İSTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

FUZZY KNAPSACK PROBLEM FOR OPERATING ROOM SCHEDULING

M.Sc. THESIS

Dilara AZAZ

Department of Management Engineering

Management Engineering Programme

JUNE 2016

İSTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

FUZZY KNAPSACK PROBLEM FOR OPERATING ROOM SCHEDULING

M.Sc. THESIS

Dilara AZAZ
(507131009)

Department of Management Engineering

Management Engineering Programme

Thesis Advisor: Prof. Dr. Ferhan ÇEBİ

JUNE 2016

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

AMELİYATHANE ÇİZELGELEME İÇİN BULANIK SIRT ÇANTASI PROBLEMİ YAKLAŞIMI

YÜKSEK LİSANS TEZİ

Dilara AZAZ
(507131009)

İşletme Mühendisliği Anabilim Dalı

İşletme Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ferhan ÇEBİ

HAZİRAN 2016

Dilara AZAZ, a M.Sc. student of ITU Graduate School of Science Engineering and Technology student ID 507131009, successfully defended the thesis entitled "FUZZY KNAPSACK PROBLEM FOR OPERATING ROOM SCHEDULING", which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Ferhan ÇEBİ** İstanbul Technical University	.....
Jury Members :	**Prof. Dr. Mehmet Mutlu YENİSEY** İstanbul University	.....
	Assoc. Prof. Dr. Sezi ÇEVİK ONAR İstanbul Technical University	.....

Date of Submission : 2 May 2016
Date of Defense : 6 June 2016

To my family,

FOREWORD

A year ago if somebody asked me what fuzzy logic was I would give a simple answer like let's say numbers higher than 1 were big while less than 1 were small. It made 1,00..1 big while 0,99..9 small. Was there really a certain line between them? That is the logic, but it would be a memorized answer. If someone asked me what were the real world applications, how it was used I wouldn't be able to answer.

Through this study I learnt much about fuzzy logic. Instead of giving a memorized answer I can explain what it is, why it is used. Also combining it with knapsack problem helped me to improve myself more. I feel thankful to my adviser Professor ÇEBİ for suggesting me this topic which let me improve myself more than I thought, and for guiding me.

I am also thankful to Beyza and Güler TOPÇU who helped me at the most critical time, and to my family and Fatih KARATAŞ who gave me their continuous support.

June 2016

Dilara AZAZ
(Industrial Engineer)

TABLE OF CONTENTS

Page

ABBREVIATIONS

CAT	**:** Computerized Axial Tomography
CDD	**:** Constrained Decision Defuzzification
COA	**:** Center of Area
COG	**:** Center of Gravity
CR	**:** Critical Ratio
DM	**:** Decision Maker
DP	**:** Dynamic Programming
EDD	**:** Earliest Due Date
ECOA	**:** Extended Center of Area
FIFO	**:** First in First out
FKP	**:** Fuzzy Knapsack Problem
FOM	**:** First of Maxima
KP	**:** Knapsack Problem
LOM	**:** Last of Maxima
LPR	**:** Linear Programming Relaxation
MOM	**:** Middle of Maxima
PI	**:** Possibility Index
RCOM	**:** Random Choice of Maxima
SPT	**:** Shortest Processing Time
S/RO	**:** Slack per Remaining Operations

SYMBOLS

$\mu\widetilde{A}(x)$, $\mu\widetilde{B}(x)$, $\mu\widetilde{D}(x)$: Membership functions of fuzzy numbers $\widetilde{A}$, $\widetilde{B}$, and $\widetilde{D}$

LIST OF TABLES

Page

LIST OF FIGURES

Page

FUZZY KNAPSACK PROBLEM FOR OPERATING ROOM SCHEDULING

SUMMARY

Scheduling is a technique whose main aim is optimizing the usage of resources. It has a broad range of application. Weekly timetabling, production scheduling, and construction scheduling can be given as examples. This thesis is about fuzzy knapsack problem, and its application on operating room scheduling.

This study consists of six parts. First part is introduction where general information about scheduling, knapsack problem, fuzzy logic, and its solution methods are given. Purpose of the thesis and its scope can also be found in introduction part.

In second part detailed information about scheduling is presented. This part starts with definition of scheduling. Explanation of a basic complex scheduling problem called Resource-Constrained Project Scheduling Problem follows definiton. For better understanding an example is given. Possible objective functions and general methods for scheduling and sequencing is presented. Then more information about patient scheduling given.

Third part is reserved for knapsack problem. It starts with definition. Then some algorithms to solve knapsack problem are presented. From the algorithms given dynamic programming is used to solve knapsack problem. For dynamic programming brief background information and detailed information about multistage problem solving is given in this part.

Fuzzy logic plays an important role in this thesis, so in fourth part fuzzy logic is explained. Solution methods for fuzzy logic follow the explanation. In the thesis solution method used for fuzziness is possibility index. Therefore, it is explained in detail.

Application is in fifth part. This study is conducted at a hospital; information about the hospital and its operating rooms is given. Then model of the problem and algorithm written with GAMS program is presented. Method used to determine fuzzy weights of surgery times and results of the application are given. Also in this part comparison of the study with other articles can be found.

In sixth part, conclusion and recommendations are presented.

AMELİYATHANE ÇİZELGELEME İÇİN BULANIK SIRT ÇANTASI PROBLEMİ YAKLAŞIMI

ÖZET

Bu tez çalışmasında ameliyathane çizelgeleme konusu ele alınmış, bulanık sırt çantası problemi yolu ile modellenip çözülmüştür. Çalışma altı kısımdan oluşmaktadır.

Birinci kısımda çizelgeleme, sırt çantası problemi, bulanık mantık konularında genel bilgi verilmiş; tezin amacı ve kapsamı açıklanmıştır.

İkinci kısımda çizelgeleme konusunda detaylı bilgi verilmiştir. Çizelgeleme geniş uygulama alanı olan bir iyileştirme yöntemidir. Ana hedefi eldeki kaynakları verimli şekilde kullanarak ihtiyaçlara cevap vermektir. Çizelgeleme ile sınırlı zaman, bütçe, kişi sayısı ve diğer kaynaklar göz önüne alınarak yapılacak işler sıraya konulur. Sıralama yapılırken işlerin aciliyeti, önemi, işlem süresi, teslim zamanı gibi unsurlar göz önüne alınır. İşler çizelgedeki sıraya göre yapılır. Uygulama alanına fabrikalardaki üretim çizelgeleri, okullardaki ders ve sınav programı çizelgeleri, hastanelerdeki randevu çizelgeleri örnek olarak verilebilir. Çizelegeleme problemini çözmek için farklı yaklaşımlar mevcuttur. Bu kısımda olası amaç fonksiyonları, problemi çözmek için kullanılan genel methotlar verilmiş, hasta çizelgeleme konusuna bir bölüm ayrılmıştır.

Üçüncü kısım sırt çantası problemine ayrılmıştır. Sırt çantası problemi yaygın olarak kullanılan bir modeldir. Burada amaç sınırlı kapasitesi olan bir çantaya bir kişinin en yüksek faydayı elde edebilecek şekilde yerleştirebileceği eşyaları seçmesidir. Kısıtlı bütçe ile yapılacak yatırım sonucu elde edilecek karı en yüksek tutmak, diyet yaparken alınabilecek sınırlı kalori miktarında en iyi şekilde doyabilmek, tarlaya ekilebilecek ürünler arasından verimli ve satışı yüksek olan ürünleri seçip onları yetiştirmek uygulama alanlarına örnek gösterilebilinir. Sırt çantası problemini çözmek için farklı algoritmalar mevcuttur. Bu kısımda algoritmalara yer verilmişitir. Tezin uygulama kısmında verilen sırt çantası probleminin çözümünde dinamik programlama yaklaşımı kullanılmıştır, burada dinamik programlama konusuna da ayrıntılı yer verilmiştir. Genel sırt çantası probleminde çanta içerisine konulacak eşyaların ağırlıkları biliniyor kabul edilir; ancak bu kullanım problemin her uygulama alanı için gerçekçi bir yaklaşım olamamaktadır. Çünkü gerçek dünyada belirsizlikler fazladır. Çantanın içine konulacak ürünlerin ağırlıkları için net bir sayı olduğu varsayımında bulunulduğunda birçok veriyi gözardı etmek gerekebilir. Bu çalışmada belirsizliğin üstesinden gelebilmek için bulanık mantıktan faydalanılmıştır.

Tezin dördüncü kısmında bulanık mantığa yer verilmiştir. Burada verilen ifade ve değerler net değildir. Bir su fabrikasında üretilen bütün bir litrelik şişelerin içindeki suların bir litre olmaması, pazardan alınan her bir kilo elmanın aslında tam bir kilo olmaması gibi örnekler bulanık mantığın kullanımı konusunda fikir verebilir. Bu kısım çözüm yöntemleri ile devam etmektedir.

Bulanık mantığın farklı çözüm yöntemleri bulunmaktadır. "Defuzzification" bulanık ağırlıkları bir miktar bilgi kaybı göze alınarak net değer haline getiren bir yöntemdir. Dördüncü kısmın içerisinde bazı defuzzification yöntemlerine yer verilmiştir. Defuzzification birçok araştırmacı tarafından da kullanılmıştır; ancak bu bulanık mantıkla ilgili soruları çözmek için tek yöntem değildir. Bazı araştırmacılar bu problemi çözmek için olasılık indeksi adı verilen bir değer hesaplatarak en iyi sonuca ulaşmaya çalışır. Olasılık indeksi eldeki kapasite ve yerleştirilmek istenen ürünün bulanık ağırlıkları bir eşitliğe dahil edilerek bulunur. Hesaplanan indexler karar vericiye bağlı olarak seçimi değiştirir. Karar verici iyimser ise ağırlığın en düşük değere yakın olacağını varsayar ve daha çok ürün koymayı tercih ederken kötümser karar verici ağırlığın en yüksek değere yakın olacağını düşünüp daha az ürün koymayı planlar. Bir de orta dereceli ya da ılımlı diye adlandırılan bir karar verici vardır ki bu karar verici dikkatli olmakla beraber olumsuza göre risk almaya daha yakındır.

Beşinci kısımda uygulamaya yer verilmiştir. Bu tez çalışmasında olasılık indeksi ile çözüme ulaşılan bulanık sırt çantası problemi üzerine çalışılmıştır. Uygulama alanı olarak da bir hastanenin ameliyathaneleri seçilmiştir.

Çalışmada üzerinde durulan temel konu belirsiz ameliyat süreleri olmuştur. Belirsiz ameliyat süreleri ile anlatılmak istenen bir kliniğin bütün hastalarının ameliyatlarının aynı sürede tamamlanamayacağıdır.

Uygulama Türkiye'deki bir eğitim ve araştırma hastanesinde gerçekleştirilmiş olup 2015 yılına ait ameliyathane verileri dikkate alınmıştır. Ameliyathane ve ameliyat olan hasta sayısının fazla olması nedeniyle çalışmanın eğitim ve araştırma hastanesinde yapılması uygun görülmüştür. Söz konusu hastanede her kliniğin kendisine ait bir ya da daha fazla ameliyathanesi bulunmaktadır. Diğer bir ifadeyle ameliyathaneler klinikler tarafından ortak kullanılmamaktadır. Bununla birlikte ameliyat süreci ameliyatların ortak özelliğini oluşturmaktadır. Her hasta hazırlanarak ameliyathaneye getirilir ve gerek duyulan hastalara anestezi uygulanır, ameliyat gerçekleştirilir ve hastalar ameliyathaneden çıkarılır. Bundan sonraki aşama ameliyathanenin bir sonraki hasta için temizlenmesidir.

Bu çalışmada ameliyat süresi hastanın ameliyathaneye getirilmesinden ameliyathanenin bir sonraki hasta için hazırlanmasına kadar geçen süreyi kapsar. Bulanık ağırlıkların belirlenmesi sırasında bazı ameliyat süreleri gözardı edilmiştir. Bunun nedeni sık gerçekleşmeyen ve normalden fazla uzun veya kısa sürmüş olan operasyonların çizelgeleme çalışmasını etkilemesini engellemektir. Bunun için önce klinik başına her bir hastanın verileri ile beşli özet oluşturulmuş, daha sonra dörde bölenler aralığı ve adım hesaplanarak dışadüşenler tespit edilmiştir. Dışa düşenler çıkarıldıktan sonra kalan verilerden ameliyat sürelerinin bulanık değerleri tespit edilebilmiştir.

Elde edilen bulanık ağırlıklar GAMS programında bu problem için yazılan kod ile çalıştırılıp karar vericinin belirlediği olasılık indeksine göre değişen en iyi sonuca ulaşılmıştır. Yazı içerisinde farklı karar vericilere göre ameliyat edilecek hasta sayısı hakkında bilgi verilmiştir.

Ekler kısmında hastane verilerinin tutulması sırasında haftalık hazırlanmış formun örneği ve uygulamanın GAMS çıktıları yer almaktadır.

1. INTRODUCTION

Scheduling operating rooms is a complex task. One must consider many aspects. Number of operating rooms, available operation time, availability and capability of doctor, emergency and specific health conditions of patient, length of surgery, equipment in operating room since not every surgery can be made in any operating room. The list goes far more.

In order to overcome difficulties of these factors some measures were taken, like dividing operating rooms between clinics. At some hospitals each clinic has their own operating room(s) while at some hospitals operating rooms are shared between couple of clinics. With division, operations required specific equipment can be scheduled in same operating room and this maximizes the usage of equipment. Also when same/similar types of surgeries are carried out in same room preparation time for the next surgery declines. Another measure for overcoming difficulties is patient prioritization. Mostly according to emergency of patients' health conditions prioritization is made for surgeries, but if surgeries are not critical first come first served rule is applied.

For scheduling problems time is essential. Giving certain times for each procedure would make problem simple. However in real life taking certain/crisp numbers for procedures can give unreliable results. Because each procedure is unique.

In this study for operating room scheduling, knapsack model is used. It is chosen because similarity of limited capacity of knapsack and limited time available for surgeries. Main focus is imprecise operation times, since for example it cannot be said every heart surgery takes 3 hours (a crisp number). In order to overcome this problem fuzzy concept is implemented. In solution process possibility index proposed by V.P. Singh and D. Chakraborty (2015), is used. A model developed with GAMS program is used to achieve solution.

1.1 Purpose of Thesis

Purpose of the thesis is to present an approach for operating room scheduling problem while taking imprecise surgery times into account.

In this study operating room scheduling problem is solved as a knapsack problem and imprecise surgery times are implemented as fuzzy surgery times. For the solution method possibility index is used and model is solved with GAMS program. Therefore this study also aims to present to reader some information about scheduling, knapsack problem, fuzzy logic, and its solution methods.

In addition this means to see whether this model is effective for operating room scheduling problem and see required information for future research.

1.2 Scope of Thesis

This study is made based on data of an education and research hospital's operating rooms. These surgery times are of 2015. Each clinic has its own surgery room(s), so surgery rooms are not mixed. Dailiy surgery times are accepted as crisp which means each day a certain amount of time is spared for surgeries. While surgery times are taken as fuzzy numbers. In order to specify fuzzy time of surgeries data taken form the hospital is used while excluding outlier operations' times.

Also in this work emergency/life threatening cases are excluded. Therefore, urgency of each patient is assumed same.

2. SCHEDULING

2.1 Definition

Scheduling is a decision making process deals with the allocation of resources to tasks over given time periods on the purpose of optimizing one or more objectives. [1] Objectives can also be defined as performance measures. [2] These performance measures can be minimum makespan or tardiness, maximum utilization, etc.

Based on the system, scheduling is applied; resources and tasks can take many different forms. For instance if resources are runways at the airport, tasks can be take-offs and landings of airplanes[1], if resource is a cement transport vehicle, tasks can be buildings waiting for cement, if resources are doctors, tasks are patients, if the resources are cleaners at a hotel tasks will be dirty rooms, etc. While looking at the examples it can be seen that the problems addressed with scheduling ranges from simple daily tasks to complex industrial ones. If we talk about complexity, from the mid 1950s scheduling theory appeared, problems addressed became closer to the industrial applications with increasing complexity. [3]

2.2 Resource-Constrained Project Scheduling Problem

To show the logic of scheduling, resource-constrained scheduling problem is a good reference. Artigues et al. [4] and Robert et al. [5] explained this problem as follows:

Resource-Constrained Project Scheduling Problem (RCPSP) is a combinatorial optimization problem and is one of the basic complex scheduling problems. It is a general scheduling problem which can be used to model many applications. Therefore it is a good topic to understand scheduling logic. The objective is to schedule some activities over time such that scarce resource capacities are met and a certain objective function is optimized. Variables used in scheduling process are:

Activities constituting the project are identified by a set $V= \{A_0, A_1 \ldots A_n, A_{n+1}\}$. A_0 and A_{n+1} are dummy activities indicate start and end of the project. Remaining activities $A= \{A_1, A_2 \ldots A_n\}$ are real activities to be scheduled.

Durations represented by p, p_i is the duration of activity A_i; p_0 and p_{n+1} equals to 0.

Precedence relations are given by E, a set of pairs such that $(A_i, A_j) \in E$. This means activity A_i precedes by activity A_j. It can also be shown like $A_i \rightarrow A_j$. For the

consistency of precedence relationship, an assumption is made that between A_i and A_j there is no cycle. And dummy activity A_0 is the predecessor of all activities while dummy activity A_{n+1} is the successor of all activities.

Renewable resources are formalized by set $R = \{R_1 \ldots R_k\}$

Availabilities of resources presented by a vector *B*. B_k denotes the availability of R_k. R_k= 1 means resource k can process 1 activity at once, it is called disjunctive resource. If R_k is greater than 1 then it processes several activities at the same time, it is called a cumulative resource.

Demands of activities for resources are shown with *b*, b_{ik} represents amount of resource R_k used per time period during the execution of A_i.

S_i represents the start time of activity A_i. C_i denotes completion time of activity A_i. $C_i = S_i + p_i$. S_0 is a reference point for the start of the project. Here the assumption is S_0= 0. A solution is feasible if the precedence constraints (2.1) and resource constraints (2.2) are fulfilled, where $A_t = \{A_i \in A \mid S_i \leq t < S_i + p_i\}$ represents the set of non-dummy activities in process at time t.

$$S_j - S_i \geq pi \; \forall \; (A_i, A_j) \in E \tag{2.1}$$

$$\sum_{Ai \in Aj} bik \leq B_k \; \forall \; R_k \in R, \; \forall t \geq 0 \tag{2.2}$$

The makespan of a schedule S is equal to S_{n+1}, the start time of the end activity.

The above defined set A_t and constraints state that an activity cannot be interrupted once it started. This is referred to as not allowing preemption.

A short description of preemptive and non-preemptive scheduling:

- If running task can be interrupted for some time and finished later, and during that time another higher priority task can be executed then this schedule is called preemptive.
- On the other hand if a task is executed till completion and it cannot be interrupted, this is called non-preemptive. [6]

Continuation of RCPSP:

The objective is to determine starting times S_i for the activities i = 1…n in such a way that:

- At each time t the total resource demand is less than or equal to the resource availability of each resource k = 1…r,
- The given precedence constraints are fulfilled, i.e., $S_i + p_i \leq S_j$ if $i \rightarrow j$,

- The makespan C_{max} = n max i=1 {C_i} is minimized, where C_i: = S_i + p_i is assumed to be the completion time of activity i.

An example by Robert et al. [5] for RCPSP:

Consider a project with n = 4 activities, r = 2 resources with capacities R_1 = 5, R_2 = 7, a precedence relation 2 → 3 and the following data in table 2.1 shows time requirements of 4 activites.

Table 2.1: Requirements of 4 activities

i	1	2	3	4
p_i	4	3	5	8
R_{i1}	2	1	2	2
R_{i2}	3	5	2	4

Solution: Two feasible schedules as solution options are presented in figure 2.1. One of the schedules is optimal.

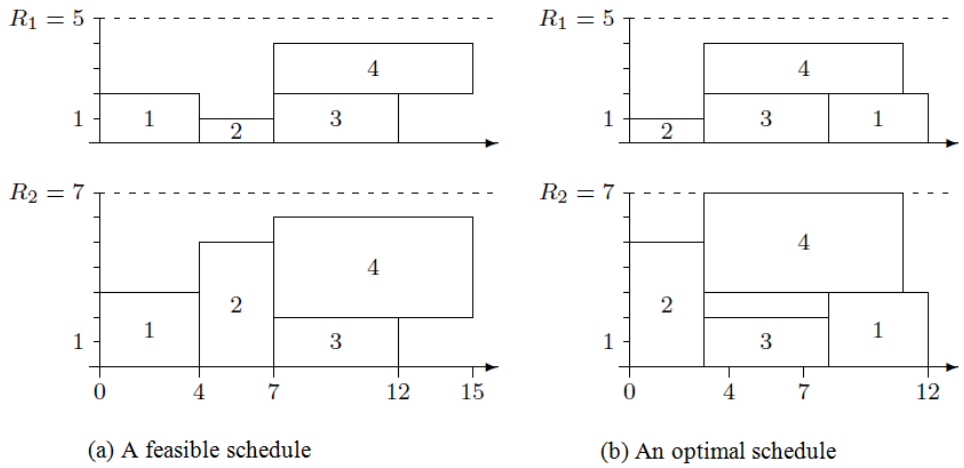


Figure 2.1: Two feasible schedules for example

C_{max} of (b) = 12 and it is an optimal schedule since it gets minimum possible makespan. On the other hand even though (a) is not optimal with C_{max} = 15 it is still feasible.

More about scheduling:

A precedence relation i → j with the meaning $S_i + p_i \leq S_j$ may be generalized by a start-start relation of the form

$$S_i + d_{ij} \leq S_j \qquad (2.3)$$

with an arbitrary integer number dij ∈ Z. The interpretation of relation (2.3) depends on the sign of d_{ij}:

- If $d_{ij} \geq 0$, then activity j cannot start before dij time units after the start of activity i. This means that activity j does not start before activity i and dij is a minimal distance (time-lag) between both starting times. Time lag mentioned here is a positive time lag.

  *$d_{ij} = p_i$ is equivalent to the precedence relation i → j.
- If $d_{ij} < 0$, then the earliest start of activity j is −dij time units before the start of activity i, i.e., activity i cannot start more than −dij time units later than activity j. If Sj ≤ Si, this means that |dij | is a maximal distance between both starting times. Time lag mentioned here is a negative time lag.

  Positive and negative time lags are shown in figure 2.2.

(a) Positive time-lag (b) Negative time-lag

Figure 2.2: Positive and negative time lags

Possible Objective Functions[4][5]

- Total flow time : $\sum_{i=1}^{n} Ci$

  Flow time is the period required for completing a specific job or a defined amount of work. [7]
- Weighted (total) flow time : $\sum_{i=1}^{n} wiCi$ with non-negative weights $w_i \geq 0$
- Due dates : d_i

  Requested time of delivery by customer for the task i.
- Lateness : $L_i := C_i - d_i$

  Lateness can be positive or negative. Positive lateness means finishing a job before its due date (earliness), while negative lateness is completing task after its due date (tardiness). Positive lateness may seem a good thing but it has some disadvantages: Customer may need that product at a specific time, and may not accept before because she has no space for it etc. So until its delivery

time producer may have to keep and it will cause holding cost. It is generally better to finish a job at a time close to its due date.

- Tardiness : $T_i := \max\{0, C_i - d_i\}$

  Tardiness shows how much time passed between completion time and due date.

- Unit penalty: $U_i := \begin{cases} 0, \text{ if } C_i \leq d_i \\ 1, \text{ otherwise} \end{cases}$

  Penalty is the cost of finishing a task after its due date.

- The maximum lateness : $L_{max} = \sum_{i=1}^{n} Li$
- The total tardiness : $\sum_{i=1}^{n} Ti$
- The total weighted tardiness : $\sum_{i=1}^{n} wiTi$
- The number of late activities : $\sum_{i=1}^{n} Ui$
- The weighted number of late activities : $\sum_{i=1}^{n} wiUi$

2.3 General Methods for Scheduling and Sequencing

There are also some basic methods used for scheduling and sequencing. According to system, and decision maker's preference one can be chosen:

1. First In First Out (FIFO): This method assigns the highest priority to the task that has been waiting for the longest time in the queue. [8]

   It has advantages like being simple, for the scheduling only required data will be arrival time of tasks. It is easy, not much effort is needed to prepare this kind of schedule. Also it is seen as fair, so for example people waiting for the service in a shop will be willing to wait till their turns come.

   On the other hand this scheduling method is non-preemptive, that is, the process will run until it finishes. Because of the non-preemptive scheduling, short processes which are at the back of the queue have to wait for the long process at the front to finish. [9]

2. Earliest Due Date (EDD): Most of the tasks have their due dates. In order to satisfy customer waiting for her order, or cure a patient before her disease becomes worse etc. due dates should be met. EDD is simple and fast, also generally performs well with regards to due date, but if not, it is because the rule does not consider the job process time. On the other hand it ignores remaining work and focus on past due job. [10]

3. Shortest Processing Time (SPT): The task requiring the shortest processing time at the workstation is processed next. [11] This method minimizes average throughput time if there is one single, indivisible resource, if schedule is preemptive, and if there is no restriction on job arrival patterns. [8]

4. Slack per Remaining Operations: Slack is the difference between the time remaining until a job's due date and the total shop time remaining, including that of the operation being scheduled. A job's priority is determined by dividing the slack by the number of operations that remain, including the one being scheduled, to arrive at the slack per remaining operations (S/RO).

$$S/RO = \frac{(\text{Due date} - \text{Today's date}) - \text{Total shop time remaining}}{\text{Number of operations remaining}} \quad (2.4)$$

   The job with the lowest S/RO is scheduled next. Ties are broken in a variety of ways if two or more jobs have the same priority. One way is to arbitrarily choose one of the tied jobs for processing next. [11]

5. Critical Ratio

   The critical ratio (CR) is calculated by dividing the time remaining until a job's due date by the total shop time remaining for the job, which is defined as the setup, processing, move, and expected waiting times of all remaining operations, including the operation being scheduled. The formula is

$$CR = \frac{(\text{Due date} - \text{Today's date})}{\text{Total shop time remaining}} \quad (2.5)$$

   The difference between the due date and today's date must be in the same time units as the total shop time remaining. A ratio less than 1.0 implies that the job is behind schedule, and a ratio greater than 1.0 implies that the job is ahead of schedule. The job with the lowest CR is scheduled next. [11]

Each method has its own advantages and disadvantages depending on the field it is used. While for a bank scheduling customers in the FIFO rule is better, for a production facility completing tasks with the SPT rule can be better to prevent bottlenecks.

Every system has its own need, and priorities. And there are systems which none of these methods can meet their requirements, but these methods can give a good idea to decision maker.

Till now general information about scheduling is given. Application part of this thesis is related to hospital, so detailed information about patient scheduling is presented below.

2.4 Patient Scheduling

Scheduling is the work of assigning resources to tasks. When it comes to patient scheduling patients become tasks while medical staff, hospital facilities and equipment become resources. Patient scheduling is a critical task, because every day thousands of people visit hospitals to receive treatment, and people's lives are on the line. If no proper system was presented there would be a mess.

When we think about patient scheduling; outpatient scheduling, surgical scheduling, and inpatient admissions are three of the most important functions in the hospital. [12]

If there is no emergency outpatients usually have to make an appointment for physical examination before visiting hospitals. Doctor's daily time is divided into time slots, and if there is a free time slot, patient can make an appointment on that day and time period. After arriving hospital necessary procedures are followed then physical examination starts.

Not all patients get appointment. Some patients just go hospital to receive treatment. They are called walk-ins. They can only have examination if there is a free timeslot or if there is no show or if other patients leave earlier than expected. Therefore there is a possibility of not having examination for them. This possibility is pretty low for scheduled patients.

Some patients who get appointment neither go to hospital for examination nor cancel their appointment. This is called no show. No shows effect performance of schedule and it happens often. A realistic model should allow no shows. No shows can be seen in many schedules, for example in flights. A passenger buys tickect then does not take the flight. Airline companies keep record of passengers therefore they know average rate of no shows, and according to data they make overbookings. However for hospitals overbooking may not be a good idea, because turning down a patient even though she has appointment could cause critical problems.

Also each patient cannot have her examination on time. Sometimes preceding patient can spend longer time than expected in the doctor's room, so next patient has to wait until current patient exits. On the contrary sometimes examination can take less time than expected. If this happens next patient or a walk-in patient can have her examination earlier.

The objective of appointment scheduling is trading off the interests of physicians and patients: the patients prefer to have a short waiting time; the physician likes to have as little idle time as possible, and to finish on time. [13] Unfortunately this cannot be achieved, because there always are unexpected events.

Another point can be classification of patients. Many systems and studies assume patients are homogeneous. It means no classification is made. Classification here is not for the clinics patients visit for their diseases, but it is differentiations of patients come to the same clinic or to another service for example CAT scans. Classification means giving different time intervals for same clinic patients. It can be based on age, maybe pediatric and geriatric patients require longer times compared to adult patients. Or it can be thought that some diseases require special times like acute problem, or chronic problem. Important point is presenting a schedule for classes without losing generality. [14]

Outpatient clinics are similar to queuing systems, whose simplest case can be all scheduled patients come on time and a single doctor serves them. System becomes complicated as doctor number increases, and multiple services are considered. Also in real time it is impossible for all patients to come on time. Emergency situations may occur, doctors can be late, medical devices may not work etc. [15]

As you can see appointment time scheduling is a complex task. Now let's think about surgical scheduling.

Operating rooms are the hospital's largest cost and revenue center, so improving their utilization directly affects the performance of the hospital. [16] Though improving operating rooms' performance sounds nice, it is a hard task. There are many factors and conflicting objectives to be considered. Some of these are availability of operating rooms, availability and capability of doctors, surgery type, patient's age, health condition, equipment in operating room etc.

There are many researches for operating room scheduling. Many of them focused on prioritization and imprecise surgery times. Prioritization is inevitable for surgery scheduling, because urgency level is different for every patient. Urgency we mention

here is not life threatening cases. For the life threatening cases emergency surgery rooms are used. But for other surgeries prioritization is made.

As an example we can talk about a study which was made for deciding which criteria should be considered for prioritization. That study is made because long delays of needed treatment may involve health risks and suffering for patients, and costs for society. Therefore, much importance is attached to reducing the negative consequences of waiting lists. In the study effects of waiting for surgery on a patient are grouped into 4 categories: Physical symptoms, psychological distress, social limitations and impairments in work. People joined to study are surgeons, physicians, practitioners and former patients. Study showed that participants agree on making prioritization while physical symptoms and impairments in the work should be more important than psychological distress and social limitaitons. [17]

There are other studies for surgery times. It is critical to estimate surgery times correctly to prevent delays caused by extended surgery times and also prevent uneffective use of the room caused by shorter surgery times than expected. Some researchs think it as a crisp number with taking average time as surgery time for that clinic [18] while some researchs take that number imprecise, stochastic.[19]

Operating patients during the limited surgical time per day is similar to the knapsack problem. In this thesis operating room scheduling problem is solved as knapsack problem.

3. KNAPSACK PROBLEM

3.1 Definition

The knapsack problem is a combinatorial optimization problem, with given a set of items, each with a weight and a value/profit. It is the task of determining the number of each item to put in to reach the highest total value possible while keeping total weight less than or equal to the capacity. It gets the name knapsack from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. [20]

Knapsack shows the capacity, and capacity can be many things. It can be production capacity, there is limited capacity for producing items, so manufacturer has to choose from the items which she is capable of producing, the most profitable ones. It can be investment budget, cropland capacity, or it can be time of a student who has to prepare for final exams and has limited time. It can be used in diet programs. For a person who plans to lose weight with a diet program has to take limited calories per day so on. [21]

Let's think about the simplest possible form of a decision which is the choice between two alternatives. Such a binary decision is formulated in a quantitative model as a binary variable $x \in \{0, 1\}$ with the obvious meaning that $x = 1$ means taking the first alternative whereas $x = 0$ indicates the rejection of the first alternative and hence the selection of the second option.

This can be extended to make choice from several options for a given capacity, like project selection. A manager has to choose several projects from various options to invest.

Now let's take it one step further and think that decision maker can take several units of same item. It becomes more complex. DM needs to decide how many of each items to take.

There also can be multiple knapsacks to be filled.

A basic knapsack problem can be formulated as follows:

max $z = \sum_{i=1}^{n} x_i p_i$

subject to $\sum_{i=1}^{n} x_i w_i \leq W$ i= 1, 2…n,

The variables presented in the equation are:

- z : objective function which will give the optimal value,
- x_i : number of item i put in knapsack,
- p_i : profit/return of item i,
- w_i: weight of item i,
- W: knapsack capacity,
- n : variety of items.

There are n types of items, and we want to fill capacity W with them. Each item has a weight (w_i) and profit (p_i). And if we put x_i amount of each item, we can get optimal solution/the highest profit z.

There are couple of algorithms to solve knapsack problems.

3.2 Some Algorithms to Solve Knapsack Problem

3.2.1 Greedy algorithm

If decision maker is not an expert she can use greedy algorithm. In greedy algorithm first step is efficiency value calculation:

$$e_i = p_i/w_i$$

After calculation items are sorted by their efficiency. DM will try to put the items which have highest efficiency. These items generate highest profit while consuming the lowest amount of capacity.

Also there is Greedy-Split algorithm which stops as soon as algorithm Greedy fails for the first time to put an item into knapsack.

3.2.2 Linear programming relaxation

The original problem is derived by optimizing over all (usually nonnegative) real values, $0 \leq x_i \leq 1$, instead of only the integer values, $x_i \in \{0, 1\}$. This adds flexibility to problem.

Naturally, for a maximization problem the optimal solution value of the relaxed problem is at least as large as the original solution value obtained with integer numbers, because the set of feasible solutions for KP is a subset of the feasible solutions for the relaxed problem.

The solution of LPR can be computed in a very simple way, as LPR possesses the greedy choice propert, i.e. a global optimum can be obtained by making a series of greedy (locally optimal) choices.

The greedy choice for LPR is to pack the items in decreasing order of their efficiencies, thus getting the highest profit per each weight unit in every step. Hence, we will assume that the items are sorted based on efficiency values. The Greedy Algorithm can be used to solve LPR with minor modifications. If adding an item to the knapsack would cause an overflow of the capacity for the first time, let's say: $\sum_{i=1}^{s-1} w_i \leq W$ and $\sum_{i=1}^{s} w_i > W$. Then the execution of Greedy is stopped and the residual capacity W - $\sum_{i=1}^{s-1} w_i$ is filled by an appropriate fractional part of item s. Item s is frequently referred to as the split item s.

3.2.3 Dynamic programming

The previous (natural) approaches fill knapsack based on intuitive way, but the solutions obtained maybe far from the optimal solution. There is no obvious strategy of guessing, so starting with a small sub problem and then extending this solution iteratively until the complete problem is solved can be a good idea. In particular it is maybe useful not to deal with all n items at once but to add items iteratively to the problem and its solution. This basic idea leads to the use of the concept of dynamic programming.

This basic property of applying dynamic programming to the knapsack problem can be described as follows: Let's assume that the optimal solution of the knapsack problem was already computed for a subset of the items and all capacities up to W.. Then we add one item to this subset and check whether the optimal solution needs to be changed for the enlarged subset. This check can be done very easily by using the solutions of the knapsack problems with smaller capacity like:

A sub knapsack problem consisting item set {1…j} and a knapsack capacity D ≤ W. For j= 0…n and D= 0…W, let $KP_j(D)$ be defined as: maximize $\sum_{l=1}^{j} p_l x_l$ subject to $\sum_{l=1}^{j} w_l x_l \leq D$, $x_l \in \{0, 1\}$, l= 1…j.

To preserve this advantage we have to compute the possible change of the optimal solutions again for all capacities. This procedure of adding an item is iterated until finally all items were considered and hence the overall optimal solution is found.

3.2.4 Branch and bound

A completely different method than dynamic programming to compute an optimal solution for integer programming problems is the branch and bound approach. Instead of extending a partial solution iteratively until finally an overall optimal solution is found as in dynamic programming, a brute force algorithm might enumerate all feasible solutions and select the one with the highest objective function value. However, the number of feasible solutions may become extremely large since 2^n different solutions can be generated from n binary variables.

Assume that we wish to solve the following maximization problem: max f(x), $x \in X$ where X is a finite solution space. In the branching part, a given subset of the solution space $X' \subseteq X$ is divided into a number of smaller subsets $X_1 ... X_m$. These subsets may in principle be overlapping but their union must span the whole solution space X, thus $X_1 \cup X_2 \cup ... \cup X_m = X'$. The process is in principle repeated until each subset contains only a single feasible solution. By choosing the best of all considered solutions according to the present objective value, one is guaranteed to find a global optimum for the stated problem.

3.2.5 Approximation algorithms

When knapsack capacity, and number of items increase even for easy problems time for reaching optimal solution increases. At some cases it would be unnecessary to find an optimal solution, instead finding a good solution in a reasonable time can be better. Paying some price (like giving up on higher profit) for the reduced running time is the basis of approximation algorithms. In principle, any algorithm which computes a feasible solution of knapsack problem is an approximation algorithm generating approximate solutions. But finding a solution close to optimal is desired result.

A possible measure for this is the investigation of the expected performance of an algorithm under a probabilistic model. In particular the expected difference of the approximate from the optimal solution would be worth to be determined. Simulation studies, where an approximation algorithm is executed many times with different sets of input data generated under a stochastic model, can give a reasonable picture of the practical behaviour of an algorithm but of course only for instances following the assumptions of the data model. Therefore, simulation cannot provide a general

certificate about the quality of an approximation method but only a reasonable guess for the behaviour of the algorithm for a particular class of problem instances. In general it may happen that an approximation algorithm sometimes generates solutions with an almost optimal value but in other cases produces inferior solutions.
For this thesis knapsack problem will be solved with dynamic programming approach, so detailed information about dynamic programming is presented in the next part.

3.3 Dynamic Programming in Detail

In the Introduction to Dynamic Programming book Nemhauser, G., L.[22] presents detailed information about dynamic programming and its logic. The parts background and multistage problem solving logic mostly depend on this reference.

3.3.1 Background

The essence of dynamic programming is making decisions in stages. Even though the effect of each stage to the final result cannot be known completely it can be known partially before passing to next stage. Objective of this process is minimizing cost of the undesirable outcome.
Before performing optimization, some changes and transformations are made on variables. While these changes are made, properties of the main problem are kept fully. The transformed problem has an optimal solution as the original, but the difference in new form is optimizing the problem becomes easier.
Dynamic programming is this kind of transformation. “It takes a sequential or multistage decision process containing many interdependent variables and converts it into a series of single-stage problems, each containing only a few variables.” In transformation process The transformation is invariant in that the number of feasible solutions and the value of the objective function associated with each feasible solution is preserved. The transformation is based on the intuitively obvious principle that “an optimal set of decisions has the property that whatever the first decision is, the remaining decisions must be optimal with respect to the outcome which results from first decision.
We may say that a problem with N decision variables can be transformed into N sub problems, each containing only one decision variable. As a rule of thumb, the computations increase exponentially with the number of variables, but only linearly with the number of sub problems. Thus there can be great computational savings.

Often this savings makes the difference between insolvable problem and one requiring only a small amount of computer time.

Certain problem areas, such as inventory theory, allocation, control theory, and chemical engineering design, have been particularly fertile for dynamic programming applications. The property basic to these problems is that decisions can be calculated sequentially. The method of sequential calculation is the essence of dynamic programming.

3.3.2 Multistage problem solving

Multistage problem solving is an approach for solving complex problems. It requires splitting the main problem into sub problems and then combining the results of sub problems in order to obtain the solution of the main problem. This approach is used if a problem cannot be solved in one step. Nemhauser, G., L.[22] defined this approach as follows:

Assume some system can be described abstractly by a state vector X_0 and the system characteristics are to be changed so that it can be described by a different state vector X_N. How can it be found appropriate transformation T_N to change the system from state X_0 to X_N? Pictorially, the problem is represented by a flow diagram shown in the figure 3.1. A transformation T_N such that $X_N = T_N(X_0)$ is sought.

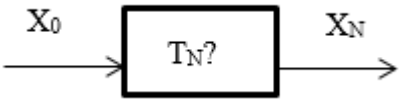


Figure 3.1: Flow diagram for multistage problem solving - 1

Suppose some transformation t_N is known which, when applied to a system in state X_{N-1}, would change the state of the system to X_{N-1} as shown in the figure 3.2. Or functionally, $X_N = t_N(X_{N-1})$.

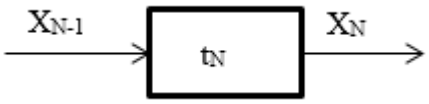


Figure 3.2: Flow diagram for multistage problem solving - 2

To solve the original problem it is only needed to find a transformation that will change the system from X_0 to X_{N-1}. Suppose T_{N-1} is such a transformation. Then the solution is given in figure 3.3.

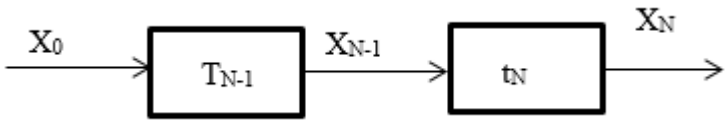


Figure3.3: Flow diagram for multistage problem solving - 3

Two boxes with respective transformation T_{N-1}, t_N taken in series are equivalent to the single box with transformation T_N. Functionally, the same equivalence is readily established since

$X_{N-1} = T_{N-1}(X_0)$ and $X_N = t_N(X_{N-1})$

together yield

$X_N = t_N(T_{N-1}(X_0)) = T_N(X_0)$

Having achieved a simplification of the problem $X_N = T_N(X_0)$ by breaking it into the two sub problems

1. $X_N = t_N(X_{N-1})$
2. $X_{N-1} = T_{N-1}(X_0)$

For proceeding further, the appropriate transformation T_{N-1} may not be apparent. But T_{N-1} can be found in the same way that T_N was found. Let's introduce another intermediate state X_{N-2} and suppose transformations T_{N-2} and t_{N-1} are known, such that $X_{N-1} = t_{N-1}(X_{N-2})$ and $X_{N-2} = T_{N-2}(X_0)$. Then by applying transformations T_{N-2}, t_{N-1}, and t_N successively to the system in state X_0, transforming it to state X_N is succeeded. Thus

$X_N = t_N(X_{N-1}) = t_N(t_{N-1}(X_{N-2})) = t_N(t_{N-1}(T_{N-2}(X_{N-2}))) = T_N(X_0)$

The corresponding flow diagram is shown in figure 3.4. The original problem has now been broken into three subproblems.

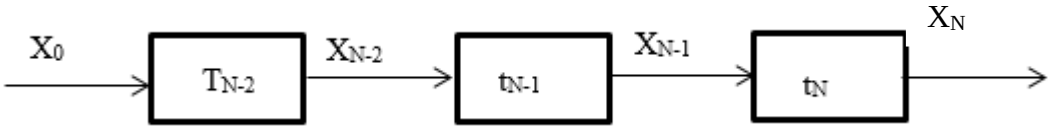


Figure 3.4: Flow diagram for multistage problem solving - 4

1. $X_N = t_N(X_{N-1})$
2. $X_{N-1} = t_{N-1}(X_{N-2})$
3. $X_{N-2} = t_{N-2}(X_0)$

To arrive at a solution it may eventually be necessary to break the original problem into N sub problems.

1. $X_N = t_N(X_{N-1})$

.

.

.

N – n. $X_{n+1} = t_{n+1}(X_n)$

N – n +1. $X_n = t_n(X_{n-1})$

.

.

.

N. $X_1 = t_1(X_0)$

Figure 3.5 shows the corresponding flow diagram.

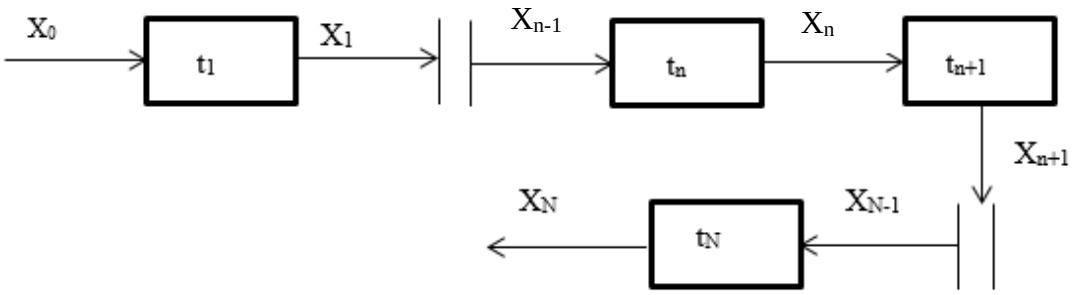


Figure 3.5: Flow diagram for multistage problem solving - 5

The multistage analysis just described started from the final state X_N and proceeded step by step with the discovery of transformations t_N, t_{N-1}…t_1, to the initial state X_0.This is called variation of multistage analysis backward multistage problem solving. If having the subscripts on the transformations agree with the order in which the transformations are determined is wanted, it is only necessary to renumber them in reverse order. To preserve the agreement between the corresponding subscripts on the transformations and states, the states are renumbered in reverse order also. Thus N becomes the initial state, and X_0 the final state. The renumbered flow diagram of figure 3.5 is shown in figure 3.6.

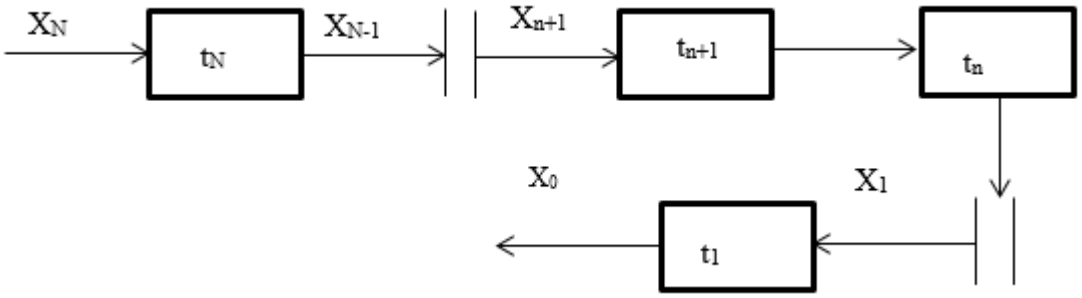


Figure 3.6: Flow diagram for multistage problem solving - 6

The corresponding N sub problems are

1. $X_0 = t_1(X_1)$

.

.

.

n. $X_{n-1} = t_n(X_n)$

n +1. $X_n = t_{n+1}(X_{n+1})$

.

.

.

N. $X_{N-1} = t_N(X_N)$

Their recursive solution yields the desired transformation T_N so that $X_0 = T_N(X_N)$. In fact,

$T_N = t_1[\ldots[t_n[t_{n+1}[\ldots[t_N]\ldots]]]\ldots]$

To see this, substitute $X_1 = t_2(X_2)$ into $X_0 = t_1(X_1)$ to obtain $X_0 = t_1[t_2(X_3)]$. Then substituting for X_2, using $X_2 = t_3(X_3$ yields

$X_0 = t_1[t_2(t_3(X_3))]$

Continuing in this manner recursively the above equation for T_N is obtained.

The determination of t_n is a stage in the step by step solution of the whole problem. In fact, finding t_n is the n^{th} stage of an N-stage process. It may be that, physically, the system actually passed through the states $X_{N-1}\ldots X_1$ in changing from X_N to X_0, or perhaps the intermediate states are merely artifacts. The really important factor, however, is that by consideration of these states, by breaking the whole problem into an equivalent series of sub problems by using multistage analysis, the problem becomes solvable.

As an alternative to backward multistage analysis we can start from the initial state X_0 and find a series of transformations leading to the final state X_N. In the forward multistage approach it is decomposed into two problems, as shown in the figure 3.7, with sub problems

1. $X_1 = t_1(X_0)$
2. $X_N = \overline{T}_{N-1}(X_1)$

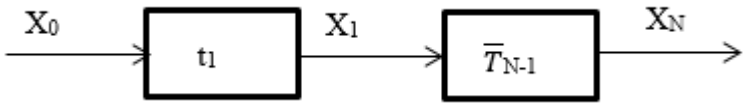


Figure 3.7: Flow diagram for multistage problem solving - 7

If the forward decomposition is carried out N-1 more times, the flow diagram of figure 3.7 is eventually reached. Conceptually, the only difference is the order in which the transformations t_n, n=1…N are discovered. However, the direction (forward or backward) of multistage analysis may make a significant difference in the ease of solving the problem. The forward approach seems more natural, but the backward approach frequently leads more easily to solution. Unfortunately, there appears to be a mental block against working backwards.

As it is mentioned many times objective is filling knapsack with some items to get the highest profit possible without exceeding capacity, but what if the weights are not known exactly. This is another point of this thesis, imprecise surgery times. In order to overcome this problem fuzzy logic is used.

4. FUZZY LOGIC

4.1 Definition

Fuzzy theory began with Zadeh, in 1965. Zadeh gave the term fuzzy to sets whose boundary is not clear, such as “a set of handsome men”, and “a set of big numbers”. He pointed out that fuzzy sets play important role in human reasoning for pattern recognition. Then he expanded it into mathematical theory. [23]

In their books Aliev, R., A.[24] and Celikyilmaz, A., and Turksen, B.[25] explained fuzzy logic as follows. Let X be a classical set of objects whose generic elements are denoted by x. Membership in a classical subset A of X is often viewed as a characteristic funtion μ_A from A to {0,1} such that $\mu_A(x) = 1$ if $x \in A$, and 0 otherwise where {0,1} is called a valuation set; 1 indicates membership while 0-non-membership.

If the valuation set is allowed to be in the real interval [0,1] then A is called a fuzzy set denoted by $\tilde{A}$, $\mu_A(x)$ is the grade of membership of x in $\tilde{A}$. It means the unique property of fuzzy sets is that, memberships in a fuzzy set are not a matter of acceptance or denial, but rather a matter of degree. Closer the value of $\mu_A(x)$ is to 1, it more belongs to $\tilde{A}$.

We can say that fuzzy system modeling has been studied to deal with complex, not clearly explained and uncertain systems, in which conventional mathematical models may fail to reveal satisfactory results. Typical membership functions are given in figure 4.1 with analytical and graphical representations.

Almost every system can be replaced with a fuzzy logic control system. This may be overkill in many places however it simplifies the design of many more complicated cases. It must be used when it is appropriate to provide better control. For example air conditioners used for cooling: Simple on-off mechanism works when the temperature drops below a present level it is turned off, while it rises above a present level it is turned on. There is a slight gap between the two present values to avoid high frequency on-off cycling. Example would be "When the temperature rises above 25 C, turn on the unit, and when temperature falls below 20 C, turn off the unit".

Using Fuzzy Rules like "If the ambient air is getting warmer, turn the cooling power up a little; if the air is getting chilly, turn the cooling power down moderately" etc. The machine will become smoother as a result of this and give more consistent comfortable room temperatures.

Criminal Search System: Helps in criminal investigation by analyzing photos of the suspects along with their characteristics like "short,young-looking.." form witnesses to determine the most likely criminals. [26]

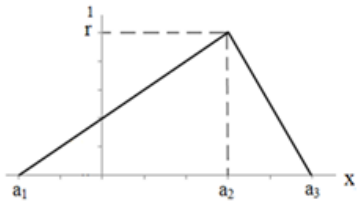
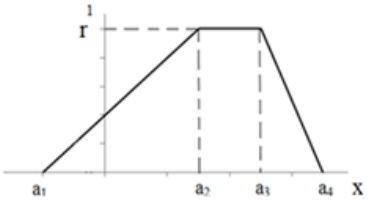
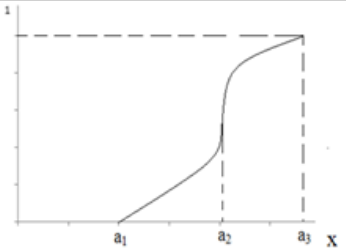
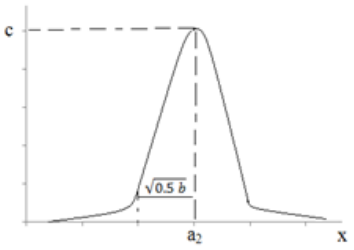
Type of Membership Function	Graphical Representation	Analytical Representation
Triangular MF		$\mu\tilde{A}(x) = \begin{cases} \frac{x-a1}{a2-a1}r, & \text{if } a_1 \le x \le a_2 \\ \frac{a3-x}{a3-a2}r, & \text{if } a_1 \le x \le a_2 \\ 0, & \text{otherwise} \end{cases}$
Trapezodial MF		$\mu\tilde{A}(x) = \begin{cases} \frac{x-a1}{a2-a1}r, & \text{if } a_1 \le x \le a_2 \\ r, & \text{if } a_2 \le x \le a_3 \\ \frac{a4-x}{a4-a3}r, & \text{if } a_3 \le x \le a_4 \\ 0, & \text{otherwise} \end{cases}$
S-Shaped MF		$\mu\tilde{A}(x) = \begin{cases} 0, & \text{if } x \le a_1 \\ 2(\frac{x-a1}{a3-a1})^2, & \text{if } a_1 < x < a_2 \\ 1-2(\frac{x-a1}{a3-a1})^2, & \text{if } a_2 \le x < a_3 \\ 0, & \text{if } a_3 \le x \end{cases}$
Bell Shaped MF		$\mu\tilde{A}(x) = c.\ \exp\left(-\frac{(x-a)}{b}^2\right)$

Figure 4.1: Typical membership functions [24]

4.2 Solution Methods

Solution of a fuzzy problem depends on decision maker. There are several solution methods some of which are listed below.

4.2.1 Defuzzification

The output of a fuzzy system is represented by fuzzy set. In applications, it is often needed to convert the output fuzzy set onto a crisp output value, which is done by defuzzification process.

In defuzzification process, several methods can be used, and selection of that method is essentially important because it affects the output value. Computational efficiency depends mostly on a kind and a number of operations required for obtaining the result of defuzzification. Basic defuzzification techniques [27] are given below:

Basic Defuzzification Techniques

The most often used defuzzification techniques are grouped according to the basic methods used in them: a group is made of the basic technique and of the all techniques extended from that basic technique. In the general case, defuzzification techniques can be formulated in a discrete form or in a continuous form. For the solution process of the thesis another technique is used so for the simplicity, only discrete form is going to be explained in this paper.

4.2.1.1 Distribution techniques

In this group of techniques, the output fuzzy set membership function is treated as a distribution, for which the average value is evaluated. Due to that heuristic approach, the output has continuous and smooth change for the change of values of input variable in the universe of discourse. The basic technique of this group is the center-of gravity technique, COG, and given by the following equation:

$$y_0 = \text{defuzzifier}(B') = \frac{\sum_{i=1}^{Nq} B'(y_i) y_i}{\sum_{i=1}^{Nq} B'(y_i)} = \text{cog}(B') \qquad (4.1)$$

where: N_q is the number of quantizing samples y_i, used in order to get the discrete form of the membership function B'(y) of the output fuzzy set B'. The calculation of the value requires $3N_q - 1$ operations which is the large number of multiplication and one division. This technique is less convenient for a hardware implementation,

because it requires large number of multipliers, as well as it requires passing through the whole universe of discourse of the output variable.

4.2.1.2 Maxima techniques

Maxima techniques give as a result of defuzzification an element from a fuzzy set core. A fuzzy set core (designated as core) consists of elements of a universe of discourse on which that set is defined with the highest degree of membership to the fuzzy set. As the basic representative of that group, the first-of-maxima technique, FOM:

$$y_0=\text{mincore}(B')=\text{fom}(B') \tag{4.2}$$

Those techniques are convenient for the general fuzzy expert systems. They are computationally efficient: they require about $2N_q$ simple operations. Maxima techniques belong to the group of the fastest defuzzification techniques, because they require passing through values of the core, only. According to the element with the maximal membership which is extracted as the defuzzification result, there are also the following maxima techniques: middle-of-maxima, MOM, last-of-maxima, LOM , and random-choice-of maxima, RCOM. The techniques are compatible with the max operation.

4.2.1.3 Area techniques

Area defuzzification techniques use area under the membership function to determine the defuzzification value. The center-of-area technique, COA, minimizes the following expression:

$$\left| \sum_{\inf y}^{coa(B')} B'(y) - \sum_{coa(B')}^{\sup y} B'(y) \right| \tag{4.3}$$

where: inf is the greatest lower bound, and sup is the least upper bound of the support of the fuzzy set B', respectively. The expression gives numerical value $y_0 = y_{coa(B')}$, which divides an area under the membership function in two (approximately) equal parts. The value ycoa(B') differs from the defuzzification value obtained by the COG technique. The calculation of the COA defuzzification value requires $4 \cdot Nq$ operations. The method is fast, because only simple operations are used in it, it gives continual change of defuzzification value; hence, it is convenient to be used in fuzzy controllers. The extended center-of-area method, ECOA, has been defined, also.

4.2.1.4 Defuzzification techniques in the decision-making systems

In the situations in which there are several output fuzzy variables, defuzzification can be considered as decision-making problem under fuzzy constraints. Then the defuzzification technique is referenced as constraint decision defuzzification, CDD. The defuzzification value - the crisp decision, under fulfilled fuzzy constraints, is obtained as a kind of a maxima technique. Such approach to defuzzification is used in. The idea of existence of various defuzzification techniques can be seen as various possibilities of ″filtering″ fuzzy output. In that way, fuzzy system designer has opportunity to satisfy different requirements. A fuzzy system is specified, by a chosen fuzzification technique, a chosen compositional rule of inference (i.e. by chosen t-norm and t conorm), and by a defuzzification technique. Different operations are chosen for the union, the intersection and the fuzzy implication will yield different results. In it has been shown that the output results depend also on hapes of membership functions.

4.2.2 Possibility index

Defuzzification techniques cause loss of information, so instead turning fuzzy result into a crisp number, possiblity index lets fuzzy weights to be kept and result to be determined based on decision maker's expertise and choice. For this solution method the possibility index created by Singh, V., P., and Chakraborty, D.[28] is presented.

PI may take any value between 0-1.

- Weight $\tilde{A}$ can be completely filled into knapsack of capacity $\tilde{B}$, if possibility index is 1.
- Weight $\tilde{A}$ cannot be filled into the knapsack of capacity $\tilde{B}$, if possibility index is 0.
- Weight $\tilde{A}$ can be filled with some possibility into knapsack of capacity $\tilde{B}$, possiblity index lies between 0 and 1.

There are 3 types of decision makers; optimistic, moderate, pessimistic. For example if decision maker is pessimistic she thinks that actual weights will be closer to highest value so she will choose an index closer to 1. In the paper[28] selection of PI based on decision maker is given in figure 4.2.

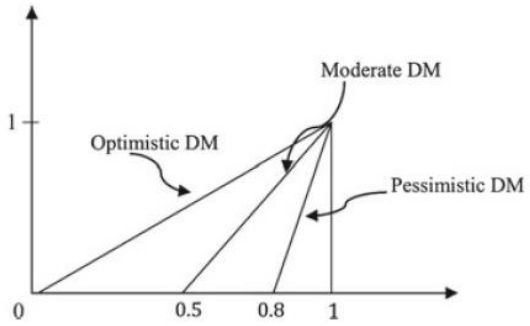


Figure 4.2: Possibility index of decision maker

4.2.2.1 PI of fuzzy weight and fuzzy capacity

Possibility index is a way to solve fuzziness of problems. Its positive point is there is no loss of information. It depends on user/decision maker's expertise.

Fuzzy weights are given as $\tilde{A}$ = (a_1, a_2, a_3), and fuzzy capacity is given as $\tilde{B}$ = (b_1, b_2, b_3). With these values PI can be calculated as:

$$PI(\tilde{A} \blacktriangle \tilde{B}) = \begin{cases} 1, & \text{if } b_3 \geq a_3 \\ 1 - y_1 * (a_3 - b_3)/(a_3 - a_1), & \text{if } b_3 < a_3 \text{ and } a_2 \leq b_2 \\ y_2 * (b_3 - a_1)/(a_3 - a_1), & \text{if } b_3 < a_3 \text{ and } a_2 > b_2 \\ 0, & \text{if } b_3 \leq a_1 \end{cases} \quad (4.4)$$

where $y_1 = \{ \mu\tilde{D}(x) \mid \mu\tilde{A}(x) = \mu\tilde{B}(x) \text{ for } x \geq b_2 \}$, $y_2 = \{ \max \mu\tilde{D}(x) \mid \mu\tilde{D}(x) = \min(\mu\tilde{A}(x), \mu\tilde{B}(x)) \}$, and $\mu\tilde{D}(x)$ represents the membership value of fuzzy set $\tilde{D} = \tilde{A} \cap \tilde{B}$.

A short explanation:

1. PI($\tilde{A} \blacktriangle \tilde{B}$) means "Area occupied by $\tilde{A}$ in the knpasack capacity $\tilde{B}$ over Total area of $\tilde{A}$.
2. y_1 equals to the membership degree $\mu\tilde{D}(x)$ whose x value gives equal membership degrees for $\mu\tilde{A}(x)$ and $\mu\tilde{B}(x)$.
3. For each x $\mu\tilde{A}(x)$ and $\mu\tilde{B}(x)$ values are calculated, and for each x value minimum of these $\mu\tilde{A}(x)$ and $\mu\tilde{B}(x)$ values are selected. The maximum of these minimum values is the y_2 value looked for.

Four sets of fuzzy numbers for fuzzy weight and fuzzy capacity is given in figure 4.3.

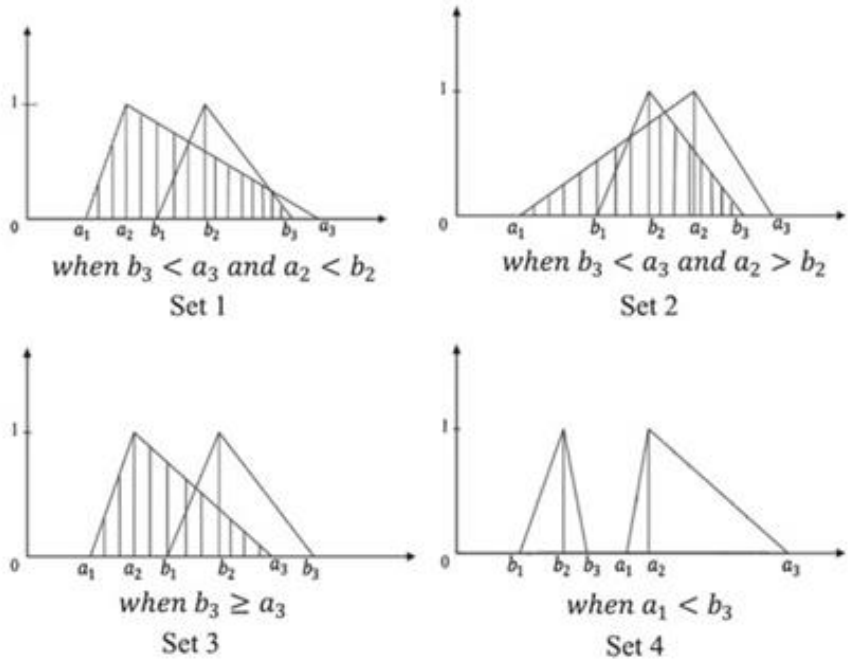


Figure 4.3: Sets of fuzzy numbers in case of fuzzy weight and fuzzy capacity

4.2.2.2 PI of fuzzy weight and crisp capacity

Fuzzy weights are given as $\tilde{A}$ = (a_1, a_2, a_3), and crisp capacity is given as b. With these values PI can be calculated as:

$$PI(\tilde{A} \blacktriangle b) = \begin{cases} 1, & \text{if } a_3 \leq b \\ 1- \mu\tilde{A}(b)^* (a_3-b)/(a_3-a_1), & \text{if } a_2 \leq b < a_3 \\ \mu\tilde{A}(b) *(b-a_1)/(a_2-a_1), & \text{if } a_1 < b < a_2 \\ 0, & \text{if } b \leq a_1 \end{cases} \quad (4.5)$$

Four sets of fuzzy numbers for fuzzy weight and crisp capacity is given in figure 4.4.

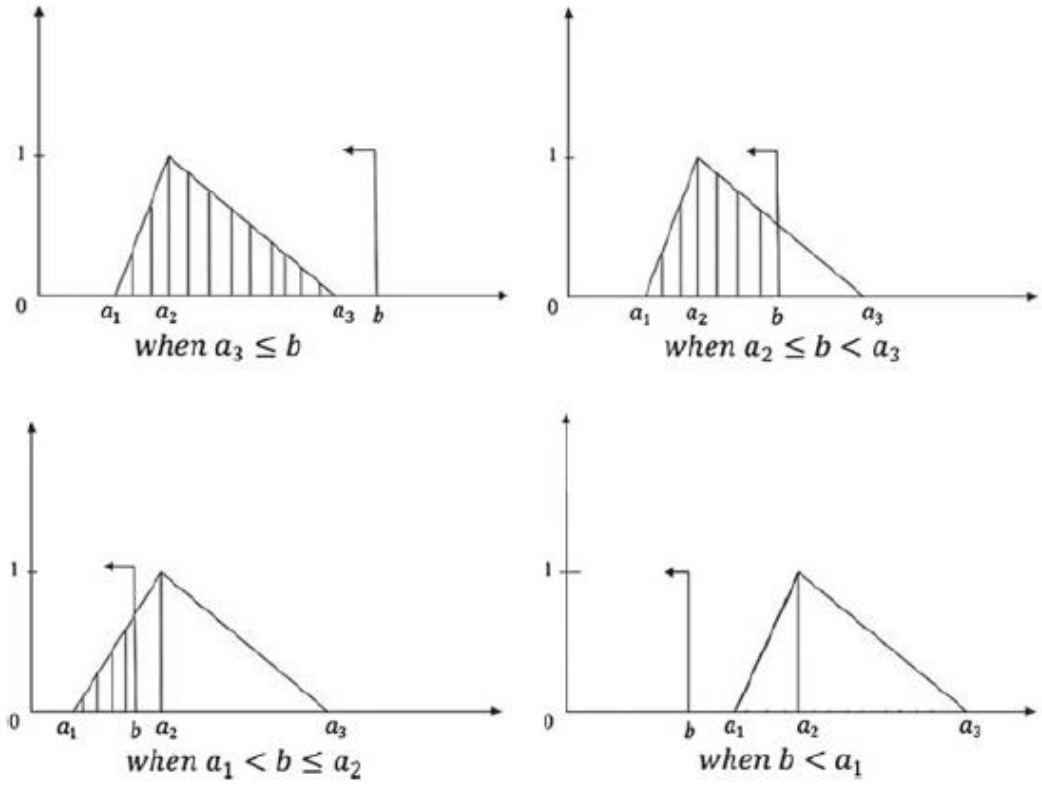


Figure 4.4: Sets of fuzzy numbers in case of fuzzy weight and crisp capacity

In this study the problem with fuzzy surgery durations is solved with PI based on fuzzy weight crisp capacity case.

5. CASE STUDY: OPERATING ROOM SCHEDULING

This study aims to propose an approach for solving operating room scheduling problem considering imprecise surgery times using fuzzy knapsack model.

Case study is carried out at an education and research hospital. Because of confidentiality name of the hospital is not given in this paper and will be addressed as the hospital.

In general, the doctors who perform surgeries make operating room scheduling. However, in terms of performance and usage of operating rooms, the hospital administration is not satisfied. That was why administration wanted personnel to keep surgery time records. Nevertheless, improvements are needed to solve this problem. The case study is carried out in order to present a new solution approach.

5.1 General Information About Hospital

Implementation of the model carried out with the data from a training and research hospital with surgical departments named in table 5.1.

Table 5.1: Surgical units

Anaesthesia and reanimation	Orthopedics and traumatology
Emergency	Otorhinolaryngology diseases
Eye diseases	Pediatric surgery
General surgery	Plastic and reconstructive surgery
Gynecology and obstetrics	Transplantation
Neurosurgery	Urology

In this hospital, doctors determine surgery schedules and 1 day before the surgery they notify responsible people of surgery rooms.

The hospital keeps record of surgery times on the purpose of keeping an eye on surgeries and finds the cause of late finished surgeries. There are digital clocks in each operating room, and every person writes down her starting and completing time of their job.

5.2 Flow of Surgery Process

With the decision of surgery, the process starts. Patient's preparation for surgery carried out then she is brought to the operating room. If it is necessary anesthesia is carried out. Then doctor comes and surgery starts. When surgery is done, patient is taken out from surgery room and the room is cleaned for the next operation. Flow diagram is given in figure 5.1.

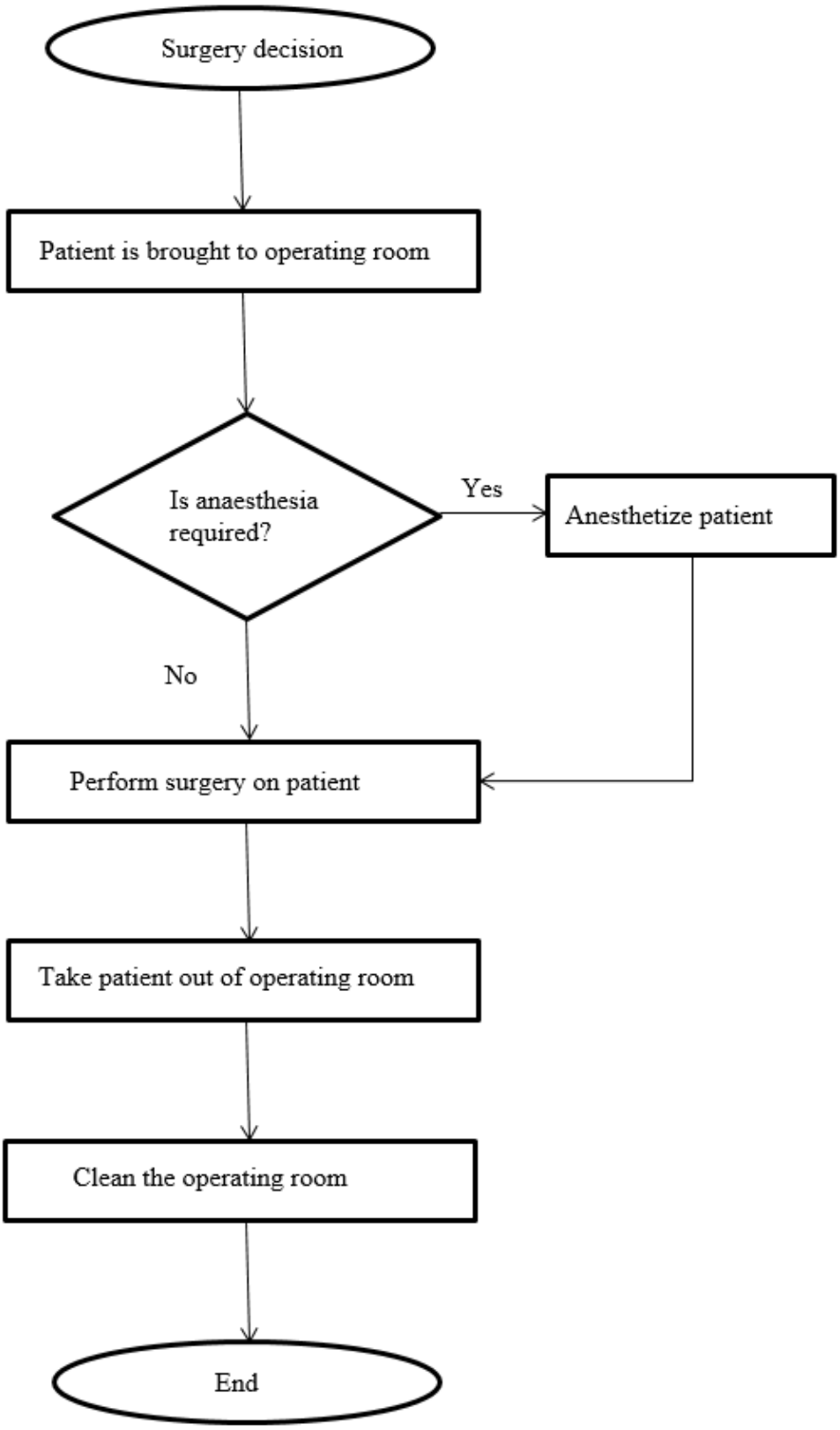


Figure 5.1: Flow diagram of surgery process

5.3 Model, GAMS Modeling and Testing

Model of the problem:

- x: number of patients
- dL: minimum surgery time of specified clinic
- dur: average surgery time of specified clinic
- dU: maximum surgery time of specified clinic
- t: daily available surgery time for operating room
- pi is greater than 0.5 for moderate decision maker
- $\mu\tilde{d}(t)$ membership function

Objective: max z = $\sum_{x=1}^{n} x$

Subject to: pi ≥ 0.5

pi = 0	if t ≤ dL*x
pi = $\mu\tilde{d}(t)$ * (t-dL*x)/(dur*x-dL*x)	if dL*x < t <dur*x
pi = 1- $\mu\tilde{d}(t)$ * (dU*x-t)/(dU*x-dur*x)	if dur*x≤ t ≤ dU*x
pi = 1	if dU*x ≤ t

dur*x ≤ t

x, dL, dur, dU, t ≥ 0

In the paper [28], an algorithm is presented, but it is not for GAMS. An algorithm is written for GAMS and verified with the values of numerical example given in the paper. After getting same results, it is used for application. Gams code is shown below.

durLow: a_1 dur: a_2 durUp: a_3 pi(j): Possibility index tM(i): Knapsack capacity

```
$Set durLow 1.5
$Set dur 2
$Set durUp 3

 Set i /f/
    j /1,2,3,4,5/;
```

Variables

```
 z;
 nonnegative variable pi(j),w(i),tM(i),x(i),y(j), u(j),n(i);
```

Equations

```
 obj, con2, Poss1, PossIn;

 Poss1..
   w.l('f') =e=0;
   tM.l('f') = 10;
   y.l('1') =1;     y.l('2') =2;     y.l('3') =3;     y.l('4') =4;     y.l('5') =5;

 Loop(j,
    if (%durLow%*y.l(j)>=tM.l('f'),
    pi.l(j)= 0;
    w.l('f')=0;

    else if ((%dur%*y.l(j)<= tM.l('f') and (%durUp%*y.l(j)> tM.l('f'))),
    pi.l(j)= 1- (%durUp%*y.l(j)-tM.l('f'))/(%durUp%*y.l(j)-
%dur%*y.l(j))*((%durUp%*y.l(j)-tM.l('f'))/(%durUp%*y.l(j)-%durLow%*y.l(j)));
    w.l('f')=0;

    else if ((%durLow%*y.l(j)<tM.l('f') and (%dur%*y.l(j)>tM.l('f'))),
    pi.l(j)= (%durUp%*y.l(j)-tM.l('f'))/(%durUp%*y.l(j)-%dur%*y.l(j))*((tM.l('f')-
%durLow%*y.l(j))/(%durUp%*y.l(j)-%durLow%*y.l(j)));
    w.l('f')=0;

    else
    pi.l(j)=1;
    w.l('f')=0;
 );  );  );  );

 PossIn..
   n.l('f')=g=0;
   scalar b;
   b=0;
 loop(j,
    if(pi.l(j)>=0.5,
    n.l('f')=y.l(j);
    b=b+1
   else b=b;
); );

 obj..    z=e= sum(i, x(i)*1);

 con2.. x('f')=l=b;

 model canta /all/;
 solve canta using mip maximizing z;
```

display z.l,x.l,pi.l,u.l,b;

5.4 Information About Data Retrieved

Thirty-seven surgery rooms' data of the year 2015 obtained from hospital. In the tables for each patient times of the operation steps presented. These are time of patient's entrance to operating room, time of anaesthesia, surgery starting time, surgery ending time, cleaning time.

From the whole data, specifying three weights is crucial. Result obtained by taking the lowest time as a_1 (minimum fuzzy weight of triangular fuzzy numbers) and the highest time as a_3 (maximum fuzzy weight of triangular fuzzy numbers) would be unreliable because there can be specific cases whose surgery times are a lot longer or shorter from the general data. Therefore, in this study outliers are calculated and excluded from the data. After outliers are excluded the lowest value remained becomes a_1, and the highest remaining value becomes a_3. a_2 is the average of all values. These are the three fuzzy weights. This elimination also means that this study focuses on elective surgeries that occur quite frequent and do not focus on elective surgeries that occur quite seldom.

Analysis results of surgery rooms based on clinics are given below:

5.5 Results

5.5.1 Gynecology and obstetrics

Gynecology and obstetrics has 1 surgery room. In 2015, 725 operations carried out in that room. Based on the data outliers are obtained and eliminated. In the calculation and elimination process values obtained are given in tables 5.2 and 5.3.

Table 5.2: 5 Points of summary for gynecology and obstatrics

09:45	Highest value
02:10	3rd quartile
01:40	2nd quartile
01:10	1st quartile
00:10	Lowest value

Table 5.3: IQR and thresholds of gynecologhy and obstatrics

01:00	Interquartile range (Q3-Q1)
01:30	1 step (IQR*1.5)
00:00	lower threshold (1st quartile - 1 step)
03:40	upper threshold (3rd quartile+ 1 step)

a_1 = 00:10 = 10 minutes

a_3 = 03:40 = 220 minutes

a_2 = 1314:29 / 725 = 01:48 = 108 minutes

683 out of 725 values fall into range [00:10;03:40]

For 1 day between 07:30 am, 04:30 pm 9 hours, 540 minutes, surgery time is given. After fuzzy surgery times are calculated they are put in GAMS model and results are obtained.

There are 3 types of decision makers: optimistic, moderate, and pessimistic. Pessimistic DM's PI is between 0.8 and 1, while moderate is between 0.5 and 0.8, and optimistic is between 0 and 0.5 (V.P. Singh and D. Chakraborty, 2015).

Now with model results for 3 types of DM are obtained and number of patients scheduled based on different possibility indexes are given in figure 5.2.

```
----     75 VARIABLE pi.L

1  1.000,    2  1.000,    3  0.932,    4  0.693,    5  0.467,    6  0.442
7  0.408,    8  0.373,    9  0.340,    10 0.311,    11 0.284,    12 0.260
```

Figure 5.2: PI's of gynecology and obstatrics

For a pessimistic DM 3 patients should be scheduled for surgery, while for optimistic decision maker 12 can be scheduled even more. Moderate says 4-5 patients per day. This is the general idea we get from looking at all PI's, but if we enter a specific value as possibility index, model will directly give the number of patients to be operated. That is the point of using possibility indexes. DM knows the doctor who is going to perform surgery. According to the data collected, DM knows how is the performance of doctor and situations of patients. She can choose PI based on this

knowledge.In order to give the idea of this method more application is presented below.

5.5.2 Orthopedics and traumatology

Orthopedics has two surgery rooms. In 2015, 637 operations carried out in those rooms. In the calculation and elimination process values obtained are given in tables 5.4 and 5.5.

Table 5.4: 5 Points of summary for orthopedics and traumatology

10:05	Highest value
02:45	3rd quartile
02:17	2nd quartile
01:35	1st quartile
00:20	Lowest value

Table 5.5: IQR and thresholds of orthopedics and traumatology

01:10	Interquartile range (Q3-Q1)
01:45	1 step (IQR*1.5)
00:00	lower threshold (1st quartile - 1 step)
04:30	upper threshold (3rd quartile+ 1 step)

a_1 = 00:20 = 20 minutes

a_3 = 04:30 = 270 minutes

a_2 = 1459:10 / 637 = 02:17 = 137 minutes

611 out of 637 values fall into range [00:20;04:30]

For 2 rooms of orthopedics 1080 minutes, surgery time is given. After fuzzy surgery times are calculated they are put in GAMS model and results are obtained. Number of patients scheduled based on different possibility indexes are given in figure 5.3.

```
----     75 VARIABLE pi.L

1  1.000,    2  1.000,    3  1.000,    4  1.000,    5  0.912,    6  0.756
7  0.597,    8  0.467,    9  0.451,    10 0.429,    11 0.404,    12 0.379
```

Figure 5.3: PI's of orthopedics and traumatology

For a pessimistic DM 5 patients should be scheduled for surgery, moderate DM can go up to 7-8 patients per day.

5.5.3 Urology

Urology has two surgery rooms. In 2015, 784 operations carried out in those rooms. In the calculation and elimination process values obtained are given in tables 5.6 and 5.7.

Table 5.6: 5 Points of summary for urology

11:20	Highest value
03:25	3rd quartile
02:46	2nd quartile
01:50	1st quartile
00:15	Lowest value

Table 5.7: IQR and thresholds of urology

01:35	Interquartile range (Q3-Q1)
02:22	1 step (IQR*1.5)
00:00	lower threshold (1st quartile - 1 step)
05:47	upper threshold (3rd quartile+ 1 step)

a_1 = 00:15 = 15 minutes

a_3 = 05:47 = 347 minutes

a_2 = 2169:24 / 784 = 02:46 = 168 minutes

748 out of 784 values fall into range [00:15;05:47]

For 2 rooms of orthopedics 1080 minutes, surgery time is given. After fuzzy surgery times are calculated they are put in GAMS model and results are obtained.

Number of patients scheduled based on different possibility indexes are given in figure 5.4.

```
----     75 VARIABLE pi.L

1  1.000,    2  1.000,    3  1.000,    4  0.900,    5  0.711,    6  0.531
7  0.452,    8  0.428,    9  0.401,    10 0.374,    11 0.348,    12 0.324
```

Figure 5.4: PI's of urology

For a pessimistic DM 4 patients should be scheduled for surgery, moderate DM can go up to 6-7 patients per day.

5.5.4 Otorhinolaryngology diseases

Otorhinolaryngology diseases have two surgery rooms. In 2015, 1489 operations carried out in those rooms. In the calculation and elimination process values obtained are given in tables 5.8 and 5.9.

Table 5.8: 5 points of summary for otorhinolaryngology

10:00	Highest value
03:00	3rd quartile
02:28	2nd quartile
01:35	1st quartile
00:14	Lowest value

Table 5.9: IQR and thresholds of otorhinolaryngology

01:25	Interquartile range (Q3-Q1)
02:07	1 step (IQR*1.5)
00:14	lower threshold (1st quartile - 1 step)
05:07	upper threshold (3rd quartile+ 1 step)

a_1 = 00:14 = 14 minutes

a_3 = 05:07 = 307 minutes

a_2 = 3681:00 / 1489 = 02:28 = 148 minutes

1395 out of 1489 values fall into range [00:14;05:07]

For 2 rooms of orthopedics 1080 minutes, surgery time is given. After fuzzy surgery times are calculated they are put in GAMS model and results are obtained.

Number of patients scheduled based on different possibility indexes are given in figure 5.5.

```
----     75 VARIABLE pi.L

1  1.000,    2  1.000,    3  1.000,    4  0.971,    5  0.822,    6  0.654
7  0.499,    8  0.447,    9  0.425,    10 0.402,    11 0.377,    12 0.354
```

Figure 5.5: PI's of otorhinolaryngology

For a pessimistic DM 5 patients should be scheduled for surgery, moderate DM can go up to 6-7 patients per day.

This model lets decision maker to see risks and chances instead of giving just 1 answer. If DM is expert enough she will be willing to take risks at some points. Also according to implementation field different constraints can be added.

5.6 Comparison of Thesis with Articles

In table 5.10 comparison of the study with some articles are given.

Table 5.10: Comparison

COMPARISON		
Article	**Disadvantage of Article's Method**	**Advantage of the Thesis' Method**
A Mixed Integer Programming Approach For Scheduling Of Operating Rooms[18]	An average surgery time is calculated for each clinic, a crisp value is used for each surgery.	Uncertian surgery times are taken into consideration, fuzzy times are used.
A stochastic optimization and simulation approach for scheduling operating rooms and recovery beds in an orthopedic surgery department[29]	For 1 clinic for each patient a surgery time is calculated and 85th percentile of surgery times is taken into consideration in order to improve utilization of operating rooms. It is far from generalization. Surgeon still specifies a surgery time for each patient.	While taking imprecise surgery times taking into consideration, a generalization is made. With this surgeon doesn't specify surgery time for each patient.
A two level metaheuristic for the operating room scheduling and assignment problem[30]	In order to schedule operating rooms authors divided daily operating times into equal time blocks for specific clinics and placed patients into this time blocks. Patients get ready according to these time blocks. Even if patients belong to same clinic surgery times vary and equal time blocks cannot met this problem.	Operating room time is not divided into time blocks, when a patient's surgery is completed next patient goes in. This increases patient's waiting time but increases utility of OR. And because of a suitable number of patients is scheduled their waiting time is not long.

6. CONCLUSION AND RECOMMENDATIONS

In this thesis, I applied a fuzzy knapsack algorithm to operating room scheduling problem in order to find the most suitable number of patients to be operated per day for each clinic. The originality of the study is solving fuzzy knapsack problem with possibility index for operating room scheduling. Fuzzy surgery times take into account characteristics like patients' health conditions, surgery room facilities which cause variations between surgeries.

In the application process, I chose an education and research hospital, the reason is it has many surgery rooms and patients to be operated. I used the data of 2015 for four clinics and calculated fuzzy surgery times with this data. In order to eliminate rarely occurred surgeries, which had too long or short surgery times, outliers were calculated. After elimination I calculated three fuzzy surgery times for each clinic which constitute three weights of triangular fuzzy numbers. With the calculation of three fuzzy surgery times model was used and alternative schedules were presented.

Normally doctors make schedules, and hospital administration is unsatisfied with performance of operating rooms. In order to use the capacity of operating rooms at maximum surgeries can be performed one after another. However, in order to do this, patients to be operated need to be ready for surgeries based on minimum required surgery time, and unless surgeries are finished in the minimum amount of time, patients have to wait. Also at most of the days some patients scheduled for the day will be delayed to next day. That is why specifying the number of patients is critical.

This model enables an administrator to make a schedule. It actually takes some control from doctors and gives some power to management, which adds a positive point for performance. Since it specifies a reasonable amount of patients to be scheduled, balance is acquired between waiting times and performance of operating rooms.

In the study emergency cases were excluded and urgency of patients to be scheduled were assumed same. For the further studies different patient groups can be specified

in the same clinic based on severity of their diseases. Also the case of clinics sharing same operating rooms can be taken into consideration.

REFERENCES

[1] **Pinedo, M., L.** (2010). Scheduling: Theory, Algorithms, and Systems. New York, NY: Springer

[2] **Leung, J., Y., T.** (2004) Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Boca Raton, FL: Chapman & Hall/CRC

[3] **T'kindt, V., Billaut, J., C.** (2002). Multicriteria Scheduling Theory (H. Scott, Trans.). Berlin, Heidelberg: Springer Berlin Heidelberg

[4] **Artigues, C., Demassey, S., and Néron, E.** (2008) Resource-Constrained Project Scheduling. London; Hoboken, New Jersey: Wiley, Iste

[5] **Robert, Y., and Vivien, F.** (2009) Introduction to Scheduling. Boca Raton, FL: CRC Press

[6] **What is pre-emptive and non-preemptive scheduling?** (2016). Retrieved May 1, 2016, from http://www.careerride.com/OS-preemptive-scheduling.aspx

[7] **Flow Time Definition** (2016). Retrieved May 1, 2016, from: http://www.businessdictionary.com/definition/flow-time.html#ixzz47h3NKSaL

[8] **Baudin, M**. (1990) Manufacturing Systems Analysis: With Application to Production Scheduling. Englewood Cliffs, NJ: Yourdon Press

[9] **Operating System Design/ Scheduling Process/ FCFS** (2016). Retrieved April 30, 2016, from https://en.wikibooks.org/wiki/Operating_System_Design/Scheduling_Processes/FCFS

[10] **Systems Production Planning** (2016). Retrieved April 30, 2016, from https://www.google.com.tr/url?sa=t&rct=j&q=&esrc=s&source=webcd=1&cad=rja&uact=8&ved=0ahUKEwjTiLmDkb7MAhXFKJoKHb4HA6IQFggbMAA&url=http%3A%2F%2Fwww.sba.oakland.edu%2FFaculty%2FFliedner%2FPOM%2520521%2FPAC.doc&usg=AFQjCNERGv2wzCXQTZj6GczqTPwFsk7NOw&sig2=N4GcA_Q6JJ-3ST8otqfikw

[11] **Operations Scheduling** (2016). Retrieved April 30, 2016, from http://wps.prenhall.com/wps/media/objects/7117/7288732/krm9e_SuppJ.pdf

[12] **Hancock, W., M. and Isken, M. W.** (1992) Patient-Scheduling Methodologies Journal of the Society for Health Systems, 3(4):83-94

[13] **Guido, C., K., and Koole, G.** (2007) Optimal outpatient appointment scheduling. Health Care Manage Sci. 10:217-229

[14] **Cayirli, T., Veral, E., and Rosen, H.** (2006) Designing appointment scheduling systems for ambulatory care services. Health Care Manage Sci 9: 47–58

[15] **Cayirli, T., and Veral, E.** (2003) Outpatient scheduling in health care: A review of literature. Production and Operations Management Vol. 12, No. 4

[16] **Cardoen, B., Demeulemeester, E., and Beilen, J.** (2010) Operating room planning and scheduling: a literature review. Europeran Journal of Operational Research, Vol. 201, Issue 3: 921-932

[17] **Oudhoff, J. P., Timmermans, D., R., M., Knol, D., L., Bijnen, A., B., and Van der Wal, G.** (2007) Prioritising patients on surgical waiting lists: A conjoint analysis study on the priority judgements of patients, surgeons, occupational physicians, and general practitioners. Social Science & Medicine 64 1863–1875

[18] **Cekic, B.** (2015) A mixed integer programming approach for scheduling of operating rooms. Verimlilik Dergisi sayı 2

[19] **Saadouli, H., Jerbi, B., Dammak, A., Masmoudi, L., and Bouaziz, A.** (2015) A stochastic optimization and simulation approach for scheduling operating rooms and recovery beds in an orthopedic surgery department. Computers & Industrial Engineering 80, 72–79

[20] **Knapsack Problem** (2016). Retrieved April 30, from https://en.wikipedia.org/wiki/Knapsack_problem

[21] **Kellerer, H.** (2004). Knapsack Problems. Berlin, Heidelberg: Springer

[22] **Nemhauser, G., L.** (1966). Introduction to Dynamic Programming, Introduction (1-2) and Basic Theory (14-18). New York, NY: Wiley

[23] **Mukaidono, M.** (2001). Fuzzy Logic for Beginners. Singapore ; River Edge, NJ : World Scientific

[24] **Aliev, R., A.** (2013). Fundamentals of Fuzzy Logic-Based Generalized Theory of Decisions. Berlin, Heidelberg : Springer Berlin Heidelberg

[25] **Celikyilmaz, A., and Turksen, B.** (2010). Modeling Uncertainty with Fuzzy Logic. Berlin: Springer

[26] **What are good real world examples of fuzzy logic being used?** (2016). Retrieved May 1, from https://www.quora.com/What-are-good-real-world-examples-of-fuzzy-logic-being-used

[27] **Saletic, D., Z., Velasevic, D., M., and Mastorakis, N., E.** Analysis of Basic Defuzzification Techniques

[28] **Singh, V., P., and Chakraborty, D.** (2015) A Dynamic Programming Algorithm for Solving Bi-Objective Fuzzy Knapsack Problem. Springer India

[29] **Saadouli, H., Jerbi, B., Dammak, A., Masmoudi, L., and Bouaziz, A.** (2015) A Stochastic Optimization and Simulation Approach for Scheduling Operating Rooms and Recovery Beds in an Orthopedic Surgery Department. Elsevier: Computers & Industrial Engineering 80:72–79

[30] **Aringhieri, R., Landa, P., Soriano, P., Tanfani, E., and Testi, A.** (2015) A Two Level Metaheuristic for the Operating Room Scheduling and Assignment Problem. Elsevier: Computer & Operations Research 54: 21-34

APPENDICES

APPENDIX A: Representative Weekly Surgery Room Observation Form

APPENDIX B: Gams Result – Gynecology and Obstetrics

APPENDIX C: Gams Result – Orthopedics

APPENDIX D: Gams Result – Urology

APPENDIX E: Gams Result – Otorhinolaryngology

APPENDIX A: Representative Weekly Surgery Room Observation Form

Table A.1: Representative observation sheet

DATE	1st Case	2nd Case	3rd Case	4th Case	5th Case	6th case
Patient entrance						
Anaesthesia						
Surgery start						
Patient exit						
Cleaning						

DATE	1st Case	2nd Case	3rd Case	4th Case	5th Case	6th case
Patient entrance						
Anaesthesia						
Surgery start						
Patient exit						
Cleaning						

DATE	1st Case	2nd Case	3rd Case	4th Case	5th Case	6th case
Patient entrance						
Anaesthesia						
Surgery start						
Patient exit						
Cleaning						

DATE	1st Case	2nd Case	3rd Case	4th Case	5th Case	6th case
Patient entrance						
Anaesthesia						
Surgery start						
Patient exit						
Cleaning						

DATE	1st Case	2nd Case	3rd Case	4th Case	5th Case	6th case
Patient entrance						
Anaesthesia						
Surgery start						
Patient exit						
Cleaning						

APPENDIX B: Gams result – Gynecology and obstetrics

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:26:15 Page 1
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m C o m p i l a t i o n

```
 4
 5   Set i /f/
 6       j /1,2,3,4,5,6,7,8,9,10,11,12/;
 7
 8   Variables
 9    z;
10    nonnegative variable pi(j),w(i),tM(i),x(i),y(j), u(j),n(i);
11
12    Equations
13    obj,  con2, Poss1,PossIn;
14
15    Poss1..
16    w.l('f') =e=0;
17    tM.l('f') = 540;
18    y.l('1') =1; y.l('2') =2; y.l('3') =3; y.l('4') =4;   y.l('5') =5;   y.l('6') =6;
19    y.l('7') =7; y.l('8') =8; y.l('9') =9; y.l('10') =10; y.l('11') =11; y.l('12') =12;
20
21     Loop(j,
22        if (10*y.l(j)>=tM.l('f'),
23        pi.l(j)= 0;      w.l('f')=0;
24        else if ((108*y.l(j)<= tM.l('f') and (220*y.l(j)> tM.l('f'))),
25        pi.l(j)= 1- (220*y.l(j)-tM.l('f'))/(220*y.l(j)-108*y.l(j))*((220*y.l
   (j)-tM.l('f'))/(220*y.l(j)-10*y.l(j)));
26        w.l('f')=0;
27        else if ((10*y.l(j)<tM.l('f') and (108*y.l(j)>tM.l('f'))),
28        pi.l(j)= (220*y.l(j)-tM.l('f'))/(220*y.l(j)-108*y.l(j))*((tM.l('f')-
   10*y.l(j))/(220*y.l(j)-10*y.l(j)));
29        w.l('f')=0;
30        else
31        pi.l(j)=1;       w.l('f')=0;   );  );  );  );
32
33    PossIn..
34    n.l('f')=g=0;
35    scalar b;
36    b=0;
37     loop(j,
38        if(pi.l(j)>0.5,  n.l('f')=y.l(j);  b=b+1
39        else b=b;  ); );
40    obj.. z=e= sum(i, x(i)*1);
41    con2.. x('f')=l=b;
42
```

```
 43    model canta /all/;
 44    solve canta using mip maximizing z;
 45
 46    display z.l,x.l,pi.l,b;
 47
```

COMPILATION TIME = 0.000 SECONDS 3 MB 24.5.6 r55090 WIN-VS8

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:26:15 Page 2
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
Equation Listing SOLVE canta Using MIP From line 44

---- obj =E= obj.. z - x(f) =E= 0 ; (LHS = 0)

---- con2 =L= con2.. x(f) =L= 4 ; (LHS = 0)

---- Poss1 =E= NONE

---- PossIn =G= NONE

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:26:15 Page 3
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
Column Listing SOLVE canta Using MIP From line 44

---- z

z
        (.LO, .L, .UP, .M = -INF, 0, +INF, 0)
     1 obj

---- x

x(f)
        (.LO, .L, .UP, .M = 0, 0, +INF, 0)
    -1 obj
     1 con2

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:26:15 Page 4
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
Model Statistics SOLVE canta Using MIP From line 44

MODEL STATISTICS

```
BLOCKS OF EQUATIONS          4     SINGLE EQUATIONS          2
BLOCKS OF VARIABLES          2     SINGLE VARIABLES          2
NON ZERO ELEMENTS            3


GENERATION TIME      =        0.000 SECONDS      4 MB  24.5.6 r55090 WIN-
VS8


EXECUTION TIME       =        0.000 SECONDS      4 MB  24.5.6 r55090 WIN-VS8

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:26:15 Page 5
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Solution Report     SOLVE canta Using MIP From line 44


               S O L V E     S U M M A R Y

     MODEL   canta            OBJECTIVE  z
     TYPE    MIP              DIRECTION  MAXIMIZE
     SOLVER  CPLEX            FROM LINE  44

**** SOLVER STATUS     1 Normal Completion
**** MODEL STATUS      1 Optimal
**** OBJECTIVE VALUE                4.0000

 RESOURCE USAGE, LIMIT          0.015      1000.000
 ITERATION COUNT, LIMIT         0    2000000000

IBM ILOG CPLEX   24.5.6 r55090 Released Nov 27, 2015 VS8 x86 32bit/MS
Windows
Cplex 12.6.2.0

Space for names approximately 0.00 Mb
Use option 'names no' to turn use of names off
LP status(1): optimal
Cplex Time: 0.00sec (det. 0.00 ticks)
Optimal solution found.
Objective :           4.000000


                    LOWER     LEVEL     UPPER    MARGINAL

---- EQU obj             .         .         .        1.000
---- EQU con2         -INF       4.000     4.000     1.000
---- EQU Poss1                (EMPTY)
---- EQU PossIn               (EMPTY)
```

```
                LOWER     LEVEL     UPPER    MARGINAL

---- VAR z          -INF      4.000     +INF       .

---- VAR x

   LOWER     LEVEL     UPPER    MARGINAL

f     .       4.000     +INF       .


**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:26:15 Page 6
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
E x e c u t i o n


----     46 VARIABLE z.L                =        4.000

----     46 VARIABLE x.L

f 4.000


----     46 VARIABLE pi.L

1  1.000,    2  1.000,    3  0.932,    4  0.693,    5  0.467,    6  0.442
7  0.408,    8  0.373,    9  0.340,    10 0.311,    11 0.284,    12 0.260


----     46 PARAMETER b                 =        4.000


EXECUTION TIME       =        0.063 SECONDS      3 MB  24.5.6 r55090 WIN-VS8


USER: GAMS Development Corporation, Washington, DC   G871201/0000CA-
ANY
      Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC0000


**** FILE SUMMARY

Input      C:\Users\Dilara\Documents\gamsdir\projdir\Untitled_31.gms
Output     C:\Users\Dilara\Documents\gamsdir\projdir\Untitled_31.lst
```

APPENDIX C: Gams result – Orthopedics

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:35:22 Page 1
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m C o m p i l a t i o n

```
 4
 5   Set i /f/
 6       j /1,2,3,4,5,6,7,8,9,10,11,12/;
 7
 8   Variables
 9    z;
10    nonnegative variable pi(j),w(i),tM(i),x(i),y(j), u(j),n(i);
11
12    Equations
13    obj,  con2, Poss1,PossIn;
14
15    Poss1..
16    w.l('f') =e=0;
17    tM.l('f') = 1080;
18    y.l('1') =1; y.l('2') =2; y.l('3') =3; y.l('4') =4;   y.l('5') =5;   y.l
   ('6') =6;
19    y.l('7') =7; y.l('8') =8; y.l('9') =9; y.l('10') =10; y.l('11') =11; y.l
   ('12') =12;
20
21    Loop(j,
22       if (20*y.l(j)>=tM.l('f'),
23       pi.l(j)= 0;      w.l('f')=0;
24       else if ((137*y.l(j)<= tM.l('f') and (270*y.l(j)> tM.l('f'))),
25       pi.l(j)= 1- (270*y.l(j)-tM.l('f'))/(270*y.l(j)-137*y.l(j))*((270*y.l
   (j)-tM.l('f'))/(270*y.l(j)-20*y.l(j)));
26       w.l('f')=0;
27       else if ((20*y.l(j)<tM.l('f') and (137*y.l(j)>tM.l('f'))),
28       pi.l(j)= (270*y.l(j)-tM.l('f'))/(270*y.l(j)-137*y.l(j))*((tM.l('f')-
   20*y.l(j))/(270*y.l(j)-20*y.l(j)));
29       w.l('f')=0;
30       else
31       pi.l(j)=1;       w.l('f')=0;   );  );  );  );
32
33    PossIn..
34    n.l('f')=g=0;
35    scalar b;
36    b=0;
37     loop(j,
38       if(pi.l(j)>0.5,  n.l('f')=y.l(j);  b=b+1
39       else b=b;  ); );
40     obj.. z=e= sum(i, x(i)*1);
41     con2.. x('f')=l=b;
42
43     model canta /all/;
```

```
 44    solve canta using mip maximizing z;
 45
 46    display z.l,x.l,pi.l,b;
 47

COMPILATION TIME     =    0.000 SECONDS      3 MB  24.5.6 r55090 WIN-VS8

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:35:22 Page 2
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Equation Listing    SOLVE canta Using MIP From line 44


---- obj  =E=  obj..  z - x(f) =E= 0 ; (LHS = 0)


---- con2  =L=  con2..  x(f) =L= 7 ; (LHS = 0)


---- Poss1  =E=               NONE


---- PossIn  =G=               NONE

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:35:22 Page 3
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Column Listing      SOLVE canta Using MIP From line 44

---- z
z
            (.LO, .L, .UP, .M = -INF, 0, +INF, 0)
      1      obj


---- x
x(f)
            (.LO, .L, .UP, .M = 0, 0, +INF, 0)
     -1      obj
      1      con2

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:35:22 Page 4
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Model Statistics    SOLVE canta Using MIP From line 44


MODEL STATISTICS

BLOCKS OF EQUATIONS           4     SINGLE EQUATIONS           2
```

BLOCKS OF VARIABLES 2 SINGLE VARIABLES 2
NON ZERO ELEMENTS 3

GENERATION TIME = 0.016 SECONDS 4 MB 24.5.6 r55090 WIN-VS8

EXECUTION TIME = 0.016 SECONDS 4 MB 24.5.6 r55090 WIN-VS8

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:35:22 Page 5
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
Solution Report SOLVE canta Using MIP From line 44

S O L V E S U M M A R Y

MODEL canta OBJECTIVE z
TYPE MIP DIRECTION MAXIMIZE
SOLVER CPLEX FROM LINE 44

**** SOLVER STATUS 1 Normal Completion
**** MODEL STATUS 1 Optimal
**** OBJECTIVE VALUE 7.0000

RESOURCE USAGE, LIMIT 0.016 1000.000
ITERATION COUNT, LIMIT 0 2000000000

IBM ILOG CPLEX 24.5.6 r55090 Released Nov 27, 2015 VS8 x86 32bit/MS Windows
Cplex 12.6.2.0

Space for names approximately 0.00 Mb
Use option 'names no' to turn use of names off
LP status(1): optimal
Cplex Time: 0.00sec (det. 0.00 ticks)
Optimal solution found.
Objective : 7.000000

LOWER LEVEL UPPER MARGINAL

---- EQU obj . . . 1.000
---- EQU con2 -INF 7.000 7.000 1.000
---- EQU Poss1 (EMPTY)
---- EQU PossIn (EMPTY)

LOWER LEVEL UPPER MARGINAL

---- VAR z -INF 7.000 +INF .

---- VAR x

```
   LOWER     LEVEL     UPPER    MARGINAL

f     .       7.000     +INF       .


**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:35:22 Page 6
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
E x e c u t i o n


----     46 VARIABLE z.L                 =        7.000

----     46 VARIABLE x.L

f 7.000

----     46 VARIABLE pi.L

1  1.000,    2  1.000,    3  1.000,    4  1.000,    5  0.912,    6  0.756
7  0.597,    8  0.467,    9  0.451,    10 0.429,    11 0.404,    12 0.379


----     46 PARAMETER b                  =        7.000


EXECUTION TIME       =        0.000 SECONDS      3 MB  24.5.6 r55090 WIN-VS8


USER: GAMS Development Corporation, Washington, DC   G871201/0000CA-
ANY
      Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC0000


**** FILE SUMMARY

Input      C:\Users\Dilara\Documents\gamsdir\projdir\Untitled_31.gms
Output     C:\Users\Dilara\Documents\gamsdir\projdir\Untitled_31.lst
```

APPENDIX D: Gams result – Urology

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:38:22 Page 1
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m C o m p i l a t i o n

```
 4
 5   Set i /f/
 6       j /1,2,3,4,5,6,7,8,9,10,11,12/;
 7
 8   Variables
 9    z;
10    nonnegative variable pi(j),w(i),tM(i),x(i),y(j), u(j),n(i);
11
12    Equations
13    obj,  con2, Poss1,PossIn;
14
15    Poss1..
16    w.l('f') =e=0;
17    tM.l('f') = 1080;
18    y.l('1') =1; y.l('2') =2; y.l('3') =3; y.l('4') =4;   y.l('5') =5;   y.l
   ('6') =6;
19    y.l('7') =7; y.l('8') =8; y.l('9') =9; y.l('10') =10; y.l('11') =11; y.l
   ('12') =12;
20
21    Loop(j,
22       if (15*y.l(j)>=tM.l('f'),
23       pi.l(j)= 0;      w.l('f')=0;
24       else if ((168*y.l(j)<= tM.l('f') and (347*y.l(j)> tM.l('f'))),
25       pi.l(j)= 1- (347*y.l(j)-tM.l('f'))/(347*y.l(j)-168*y.l(j))*((347*y.l
   (j)-tM.l('f'))/(347*y.l(j)-15*y.l(j)));
26       w.l('f')=0;
27       else if ((15*y.l(j)<tM.l('f') and (168*y.l(j)>tM.l('f'))),
28       pi.l(j)= (347*y.l(j)-tM.l('f'))/(347*y.l(j)-168*y.l(j))*((tM.l('f')-
   15*y.l(j))/(347*y.l(j)-15*y.l(j)));
29       w.l('f')=0;
30       else
31       pi.l(j)=1;       w.l('f')=0;   );  );  );  );
32
33    PossIn..
34    n.l('f')=g=0;
35    scalar b;
36    b=0;
37     loop(j,
38       if(pi.l(j)>0.5,  n.l('f')=y.l(j);  b=b+1
39       else b=b;  ); );
40     obj.. z=e= sum(i, x(i)*1);
41     con2.. x('f')=l=b;
42
43     model canta /all/;
```

```
 44    solve canta using mip maximizing z;
 45
 46    display z.l,x.l,pi.l,b;
 47

COMPILATION TIME     =     0.000 SECONDS      3 MB  24.5.6 r55090 WIN-VS8

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:38:22 Page 2
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Equation Listing    SOLVE canta Using MIP From line 44


---- obj  =E=  obj..  z - x(f) =E= 0 ; (LHS = 0)


---- con2  =L=  con2..  x(f) =L= 6 ; (LHS = 0)


---- Poss1  =E=               NONE


---- PossIn  =G=               NONE

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:38:22 Page 3
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Column Listing      SOLVE canta Using MIP From line 44

---- z
z
            (.LO, .L, .UP, .M = -INF, 0, +INF, 0)
      1      obj

---- x
x(f)
            (.LO, .L, .UP, .M = 0, 0, +INF, 0)
     -1      obj
      1      con2

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:38:22 Page 4
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Model Statistics    SOLVE canta Using MIP From line 44


MODEL STATISTICS

BLOCKS OF EQUATIONS           4     SINGLE EQUATIONS           2
BLOCKS OF VARIABLES           2     SINGLE VARIABLES           2
```

NON ZERO ELEMENTS 3

GENERATION TIME = 0.000 SECONDS 4 MB 24.5.6 r55090 WIN-VS8

EXECUTION TIME = 0.000 SECONDS 4 MB 24.5.6 r55090 WIN-VS8

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:38:22 Page 5
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
Solution Report SOLVE canta Using MIP From line 44

S O L V E S U M M A R Y

MODEL canta OBJECTIVE z
TYPE MIP DIRECTION MAXIMIZE
SOLVER CPLEX FROM LINE 44

**** SOLVER STATUS 1 Normal Completion
**** MODEL STATUS 1 Optimal
**** OBJECTIVE VALUE 6.0000

RESOURCE USAGE, LIMIT 0.016 1000.000
ITERATION COUNT, LIMIT 0 2000000000

IBM ILOG CPLEX 24.5.6 r55090 Released Nov 27, 2015 VS8 x86 32bit/MS Windows
Cplex 12.6.2.0

Space for names approximately 0.00 Mb
Use option 'names no' to turn use of names off
LP status(1): optimal
Cplex Time: 0.00sec (det. 0.00 ticks)
Optimal solution found.
Objective : 6.000000

LOWER LEVEL UPPER MARGINAL

---- EQU obj . . . 1.000
---- EQU con2 -INF 6.000 6.000 1.000
---- EQU Poss1 (EMPTY)
---- EQU PossIn (EMPTY)

LOWER LEVEL UPPER MARGINAL

---- VAR z -INF 6.000 +INF .

---- VAR x

	LOWER	LEVEL	UPPER	MARGINAL
f	.	6.000	+INF	.

**** REPORT SUMMARY : 0 NONOPT
0 INFEASIBLE
0 UNBOUNDED

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:38:22 Page 6
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
E x e c u t i o n

---- 46 VARIABLE z.L = 6.000

---- 46 VARIABLE x.L

f 6.000

---- 46 VARIABLE pi.L

1 1.000, 2 1.000, 3 1.000, 4 0.900, 5 0.711, 6 0.531
7 0.452, 8 0.428, 9 0.401, 10 0.374, 11 0.348, 12 0.324

---- 46 PARAMETER b = 6.000

EXECUTION TIME = 0.000 SECONDS 3 MB 24.5.6 r55090 WIN-VS8

USER: GAMS Development Corporation, Washington, DC G871201/0000CA-ANY
Free Demo, 202-342-0180, sales@gams.com, www.gams.com DC0000

**** FILE SUMMARY

Input C:\Users\Dilara\Documents\gamsdir\projdir\Untitled_31.gms
Output C:\Users\Dilara\Documents\gamsdir\projdir\Untitled_31.lst

APPENDIX E: Gams result – Otorhinolaryngology

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:40:58 Page 1
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m C o m p i l a t i o n

```
 4
 5   Set i /f/
 6       j /1,2,3,4,5,6,7,8,9,10,11,12/;
 7
 8   Variables
 9    z;
10    nonnegative variable pi(j),w(i),tM(i),x(i),y(j), u(j),n(i);
11
12    Equations
13    obj,  con2, Poss1,PossIn;
14
15    Poss1..
16    w.l('f') =e=0;
17    tM.l('f') = 1080;
18    y.l('1') =1; y.l('2') =2; y.l('3') =3; y.l('4') =4;   y.l('5') =5;   y.l
   ('6') =6;
19    y.l('7') =7; y.l('8') =8; y.l('9') =9; y.l('10') =10; y.l('11') =11; y.l
   ('12') =12;
20
21     Loop(j,
22       if (14*y.l(j)>=tM.l('f'),
23       pi.l(j)= 0;      w.l('f')=0;
24       else if ((148*y.l(j)<= tM.l('f') and (307*y.l(j)> tM.l('f'))),
25       pi.l(j)= 1- (307*y.l(j)-tM.l('f'))/(307*y.l(j)-148*y.l(j))*((307*y.l
   (j)-tM.l('f'))/(307*y.l(j)-14*y.l(j)));
26       w.l('f')=0;
27       else if ((14*y.l(j)<tM.l('f') and (148*y.l(j)>tM.l('f'))),
28       pi.l(j)= (307*y.l(j)-tM.l('f'))/(307*y.l(j)-148*y.l(j))*((tM.l('f')-
   14*y.l(j))/(307*y.l(j)-14*y.l(j)));
29       w.l('f')=0;
30       else
31       pi.l(j)=1;       w.l('f')=0;   );  );  );  );
32
33    PossIn..
34    n.l('f')=g=0;
35    scalar b;
36    b=0;
37     loop(j,
38       if(pi.l(j)>0.5,  n.l('f')=y.l(j);  b=b+1
39       else b=b;  ); );
40     obj.. z=e= sum(i, x(i)*1);
41     con2.. x('f')=l=b;
42
43     model canta /all/;
```

```
 44    solve canta using mip maximizing z;
 45
 46    display z.l,x.l,pi.l,b;
 47

COMPILATION TIME     =        0.000 SECONDS      3 MB  24.5.6 r55090 WIN-VS8
```


Equation Listing   SOLVE canta Using MIP From line 44

```
---- obj  =E=  obj..  z - x(f) =E= 0 ; (LHS = 0)


---- con2  =L=  con2..  x(f) =L= 6 ; (LHS = 0)


---- Poss1  =E=               NONE


---- PossIn  =G=              NONE
```


Column Listing   SOLVE canta Using MIP From line 44

```
---- z
z
            (.LO, .L, .UP, .M = -INF, 0, +INF, 0)
      1      obj

---- x

x(f)
            (.LO, .L, .UP, .M = 0, 0, +INF, 0)
     -1      obj
      1      con2
```


Model Statistics   SOLVE canta Using MIP From line 44

MODEL STATISTICS

BLOCKS OF EQUATIONS 4 SINGLE EQUATIONS 2
BLOCKS OF VARIABLES 2 SINGLE VARIABLES 2
NON ZERO ELEMENTS 3

GENERATION TIME = 0.016 SECONDS 4 MB 24.5.6 r55090 WIN-VS8

EXECUTION TIME = 0.016 SECONDS 4 MB 24.5.6 r55090 WIN-VS8

GAMS 24.5.6 r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:40:58 Page 5
G e n e r a l A l g e b r a i c M o d e l i n g S y s t e m
Solution Report SOLVE canta Using MIP From line 44

S O L V E S U M M A R Y

MODEL canta OBJECTIVE z
TYPE MIP DIRECTION MAXIMIZE
SOLVER CPLEX FROM LINE 44

**** SOLVER STATUS 1 Normal Completion
**** MODEL STATUS 1 Optimal
**** OBJECTIVE VALUE 6.0000

RESOURCE USAGE, LIMIT 0.016 1000.000
ITERATION COUNT, LIMIT 0 2000000000

IBM ILOG CPLEX 24.5.6 r55090 Released Nov 27, 2015 VS8 x86 32bit/MS Windows
Cplex 12.6.2.0

Space for names approximately 0.00 Mb
Use option 'names no' to turn use of names off
LP status(1): optimal
Cplex Time: 0.00sec (det. 0.00 ticks)
Optimal solution found.
Objective : 6.000000

LOWER LEVEL UPPER MARGINAL

---- EQU obj . . . 1.000
---- EQU con2 -INF 6.000 6.000 1.000
---- EQU Poss1 (EMPTY)
---- EQU PossIn (EMPTY)

LOWER LEVEL UPPER MARGINAL

```
---- VAR z           -INF      6.000     +INF       .

---- VAR x

   LOWER     LEVEL     UPPER    MARGINAL

f     .       6.000     +INF       .


**** REPORT SUMMARY :       0     NONOPT
                            0 INFEASIBLE
                            0  UNBOUNDED

GAMS 24.5.6  r55090 Released Nov 27, 2015 WIN-VS8 x86 32bit/MS Windows
05/08/16 05:40:58 Page 6
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
E x e c u t i o n


----     46 VARIABLE z.L                 =        6.000

----     46 VARIABLE x.L

f 6.000


----     46 VARIABLE pi.L

1  1.000,    2  1.000,    3  1.000,    4  0.971,    5  0.822,    6  0.654
7  0.499,    8  0.447,    9  0.425,    10 0.402,    11 0.377,    12 0.354


----     46 PARAMETER b                  =        6.000


EXECUTION TIME       =        0.000 SECONDS      3 MB  24.5.6 r55090 WIN-VS8


USER: GAMS Development Corporation, Washington, DC   G871201/0000CA-
ANY
      Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC0000


**** FILE SUMMARY

Input      C:\Users\Dilara\Documents\gamsdir\projdir\Untitled_31.gms
Output     C:\Users\Dilara\Documents\gamsdir\projdir\Untitled_31.lst
```

CURRICULUM VITAE

Name Surname **:** Dilara Azaz

Place and Date of Birth **:** Gölcük / 1990

E-Mail **:** azazd@itu.edu.tr

EDUCATION

- **B.Sc.** **:** 2012, Marmara University, Engineering Faculty, Industrial Engineering Department